MV2 cryptographic algorithm

Dr. Ph. Prof. V. Michtchenko,
Dr. Ph. V. Lepin,
Y. Vilanski.

2006

Valentin Michtchenko, Victor Lepin, Yury Vilanski. MV2 cryptographic algorithm

.

NGS[1] *recommends that MV2 be published so it can undergo review from the cryptographic community. ... MV2 presents several specific and advantageous characteristics:*

1. *MV2 can complement and co-exist with well-known and proven encryption technologies; most of the commercially valuable benefits can be achieved without changing existing corporate encryption technologies and policies.*

2. *MV2 "one-time" encryption produces unique, tamperproof ciphertext even when encrypting the same plaintext repeatedly using the same key.*

3. *MV2's ability to require the presence of 3 or more files to read encrypted messages; this provides substantial flexibility and savings when moving or securing data.*

NGS Consulting

[1] NGS (Next Generation Security Software Limited) is the world leader in the discovery and publication of computer security vulnerabilities, and in this capacity work as advisers to CESG, the UK Government's National Technical Authority for Information Assurance.

Contents

Introduction

This book describes the cryptographic algorithm MV2 which is a stochastic cipher. Its most distinctive feature is that the generated ciphertext is divided into two or more parts. In addition it is possible to vary the length of those generated parts. This allows to use the MV2 algorithm:

1) as a symmetrical cryptographic algorithm with 128bit, 256bit, 512bit, 1024bit, 2048bit, 4096bit keys;

2) as a way to split a ciphertext into parts with different lengths to organize several channels for cifered data transfer.

If necessary, the key length can be extended to 53890 bit.

These distinctive features are essential in business processes where to give a positive result some control parameters are required that the MV2 is able to provide. In particular, the MV2 in combination with asymmetrical cryptographic algorithms can produce new symmetrical-asymmetrical algorithms that allow instant encoding of large volumes of information using two keys simultaneously (the one of symmetrical system and the public key of the asymmetrical system); or encode and sign the documents in a single round; or encode using two keys and sign simultaneously.

Division of ciphertexts into several channels means multichannel encryption which provides additional degrees of freedom for control parameters that can be used in various applications.

There are three rather general models of building multichannel encryption systems:

1. systems where a number of short length ciphertexts are enciphered with theoretically strong system or steganogrphically concealed.

2. systems where information is partially sent to the recipient (e.g. for storage) and the rest is not sent anywhere. In this case the keeper of the stored information cannot use it or transfer it to the third party.

3. systems where information is spatially separated by using other data channels. This increases resistance against unauthorized reading of messages.

The algorithm can be used to develop new protocols for comprehensive fraud protection systems in trade, multimedia, document management and a number of other applications.

Chapter 1
Harmed texts

1.1 Sense of texts and information theory

A French physicist L.Brillouin [6] discovered interrelation between information and physical entropy. This interrelation was laid in the very foundation of information theory when C.Shannon proposed to adapt a stochastic entropy function from statistical thermodynamics to calculate the quantity of information.

The entropy function can also be used to study a sensible text. Symbols in a sensible text have different probability of their appearance. They occur not chaotically but are organized according to the rules of word formation and usage of words in expressions. However, any texts are described not only by entropic characteristics but also by sense and value of information they contain. C.Shannon simplified his model intentionally: information theory does not consider the properties of information transferred. These properties concern only transmission and receiving parties. Shannon's information theory gives a quantitative measure of the transferred information without considering its sense and value.

Hereafter we shall use the term ”sense”. Therefore we will give an encyclopedic definition of this term.

In a ”sense – text” model the term ”sense” describes the global content of a statement.

Definition 1.1 *The term ”sense” can mean an integrated content of any statement that is not reduced to the meaning of its components but defines that meaning itself.*

Any statement contains notional words that have sense and auxiliary words. It depends on the sense of a sentence that any of its parts (subject, predicate, adverbial modifier of place and time, object, more rarely - adjective, and sometimes even preposition) become notional words. To better understand this definition we will use an example.

Example 1.1 *The phrase ”I go to the cinema” may have several senses and different key words for each of them. If the sense is who goes to the cinema then the key word is ”I” and the phrase ”go to the cinema” has no sense, etc.*

Information theory does not estimate the sense and value of information as these properties are subjective. Information theory only enables to estimate the extent to which a text is ordered or how distant it is from absolute chaos when all letters are equally likely to occur and the text is a senseless set of symbols.

The more different the probability for each letter is and the stronger intersymbol influence over adjacent letters is the more a text is ordered. In the mean time, the quantity of information representing this order will be equal to the decrease of the text entropy as compared to its maximum value representing the absence of order in the text, i.e. equal probabilities for adjacent letters. Methods suggested by C. Shannon to calculate the quantity of information enable to determine the ratio between the quantity of predictable information (i.e. that is formed according to certain rules) and of unexpected information that cannot be predicted. Shannon defined information contained in the rules as redundant because knowing the rules of making up messages allows predicting probabilities of letters before they are actually transmitted.

E.g., for the English language with the number of letters $N = 26$ this value is $R = \log 26 = 4,7$ bits per letter. It is the maximum entropy of particular symbols. But because probabilities of particular symbols are different, in reality *the language entropy per symbol* of message M is $r = H(M)/L$, where $H(M)$ means message uncertainity, and L - message length in symbols. In a number of researches [8] the value of the entropy per symbol is defined for rather long messages of the English language, which is 1,3 bits per letter.

The value $B = R - r$ is usually called *redundancy in language*. Redundancy in the English language is about 3,4 bits per letter.

In a simplified model of the English language without all punctuation marks, spaces and numbers redundancy for the 8 bit letter image in the ASCII table for $r = 1,3$ will be 6,7 bits per letter!

Encryption system allows to transform the alphabet of a source text into that of a ciphertext and vice versa without changing the sense of message. The purpose is to conceal the sense in an alphabet of the plain language. However, redundancy in language causes certain information about the source text being kept in the ciphertext. This and a number of statistical regularities allow a cryptanalyst reading ciphertexts without knowing the key or even defining this key. C.Shannon demonstrated that if an encryption system completely eliminates redundancy in a source text, it becomes in principle impossible to restore a source text from a ciphertext.

1.2 Cryptanalytic attacks. Idea of harmed texts

Consider a cipher defined as (E, E^{-1}, M, Y, K). Here M, Y mean a plaintext and a ciphertext respectively, E, E^{-1} are encryption and decryption transformations, and K is a secret key. The most common situations for cryptanalysis can be reduced to the following cases:

1. One or several ciphertexts Y are available. The objective of cryptanalysis is to define E (cipher type) and find E, E^{-1}, M.

2. One or several pairs (M, Y) are available. Define cipher type E or E^{-1} and find K.

3. Cipher type E or E^{-1} and one or several ciphertexts Y are available. Find M or K, M.

4. Cipher type E or E^{-1} and one or several pairs (M, Y) are available. Find K.

5. E, E^{-1}, one ciphertext Y or pairs (M, Y), a transformation form $E(., K)$ are kavailable, but K and $E^{-1}(., K)$ are unavailable. Find K. Such formulation is typical for public-key systems.

$$\begin{aligned} Y' &= E_1(M, K) \\ Y'' &= E_2(M, K), \end{aligned}$$

where the mappings E_1 and E_2 are not injective, but are connected as follows: there is such a mapping E_{12}^{-1}, that for any source text M and key K the following holds:

$$M = E_{12}^{-1}(Y', Y'', K).$$

Let Y'' be concealed from a cryptanalyst due to transmission via another channel, or any stehanographical methods that are unknown to a cryptanalyst, or be encrypted by a theoretically strong system if Y'' has a short length.

Apart from greater key space caused by cryptographic division, a cryptanalyst faces a task of enumeration to define the missing part of Y''. It might be a very time-consuming due to the fact that the length of Y'' may be greater than the length of K.

This idea can be developed by introducing m ciphertexts and assuming that:

$$\begin{aligned} &Y' = E_1(M, K)\\ &Y'' = E_2(M, K)\\ &\ldots\\ &Y'^{(m)} = E_m(M, K);\\ &M = E^{-1}_{1\ldots m}(Y', Y'', \ldots Y'^{(m)}, K). \end{aligned}$$

The described ciphertexts will be considered as $Y'^{(i)}$ *harmed texts*, if each of them has no sense in an alphabet of a ciphertext.

1.3 Concept of harmed texts

Let there be a message M with the length L_0 and the sense $S(M)$. Let this message be written in a language with an alphabet A, with redundancy in language B_A and respective redundancy in message

$$B(M) = B_A L_0 = \left(\log N - \frac{H(M)}{L_0}\right) \cdot L_0.$$

Let an ideal archiver be abailable that allows to eliminate all redundancy and generate M' – a message with the minimal length L_{min} without losing the sense, i.e.:

$$S(M') = S(M).$$

We note here that any archiver compresses a message by making its length shorter, but keeps its sense unchanged converting an alphabet of a plain message into an alphabet of an archiver, like an ordinary information encoder does. Actually, any archiver operates beyond the ideal. That is why a text M'' it generates has a length that is greater than L_{min} without losing the sense of the plain message:

$$S(M) = S(M). \tag{1.1}$$

It is evident that any further attempts to compress the message will cause distortion of its sense and, therefore, for any text M^* with a length $L < L_{min}$ the equality (1.1) will not be observed:

$$S(M^*) \neq S(M).$$

This effect takes place because the further compression of text doesn't occurs by eliminating redundancy in letter codes but by deformation of inredundant letter codes. Herein under deformation of letters we understand further shortening of length of letter codes *beyond their information irredundancy.*

Definition 1.2 *We will consider a text as harmed if it was generated with further deformation of irredundant letter codes.*

Thus, the necessary and sufficient condition for a text to be harmed with losing its sense is shortening the length of its letter codes beyond their irredundancy. As a result, a harmed text is shorter than a source text and does not have the sense of a source text.

It follows from the definition that the whole variaty of texts generated from a source text by a number of transformations consists of two disjoint subsets: harmed texts and texts that have the sense of a source text. Please notice that all ciphertexts have the sense of a source text in an alphabet of a ciphertext, all compressed texts have the sense of a source text, and all primitively transformed source texts can retain the sense of a source text by throwing out particular words or symbols. Therefore, all these transformed texts are not harmed.

There are many methods to examine if a text is sensible position of redundancy in language. In contrast to attempts of measuring the sense we come to the idea of mere verification of whether the sense is present ot not, possibly with measuring probability, stating

$$P(S(M)) + Q(S(M)) = 1,$$

where P and Q are respective probabilities of presence and absence of the sense. It results in the need to develop an algorithm that would harm the sense in source texts or ciphertext (harming algorithm), and where $P(S(M))$ would be close to 0. So, we shall not touch on problems of measuring the sense or defining signs of present a sense; we will be interested in *a sense destruction algorithm* with a probabilistic measure of its presence or absence.

This algorithm may have its own key which additionally increases the key space.

Definition 1.3 *A cyclic algorithm that implies random substitution of bit representation of each symbol of a source text by a tuple of a smaller or equal number of bits with their further concatenation to obtain a harmed text shall be a universal damage algorithm denoted by C_m, where m is the number of rounds.*

It follows from the definition that during the harming of text codes symbols are replaced by tuples of different length. The advantage of such approach is its universality. Irrespective of the nature of a source text (texts in natural languages, ciphertexts, texts of program files, etc.) this approach allows destroying a sense and verify its absence after the harming algorithm C_m was executed during m rounds.

As further shortening of a text beyond irredundancy leads to distortion of sense in the message, additional information is required to restore the text from a harmed text. We shall call this additional information *a harm*.

A harm restores destroyed transformation injectiveness after irregular substitutions in a given harmed text. We shall only touch on such rules of harming that do not allow for a a source text to be restored (possibly, except for attempts of enumeration) if only harmed texts or harms are available. We shall be interested in such rules of harming that require knowledge of all harmed texts, all harms, and the very rule of harming to restore a source text. This idea allows pretty flexible algorithms of implementation as the process of harming can be cyclic and the rule of harming can be modified. The length of the final harmed text can be managed at every step by changing the number of steps. This harming algorithm destroys the sense of a source text and generates information that allows restoring a source text and its sense.

A harmed text is always random as it is defined only by random tuples of variable length. A harm describes only the length of random substitutions during execution of C_m and does not bear any semantics actually being a harmed text.

As a result we have two ciphertexts (a harm and a harmed text), each of them having no sense neither in the alphabet of a source text nor in the alphabet of a ciphertext. We actually presented a ciphertext from a source text into two harmed ciphertexts that when taken separately cannot restore the source text. Here we have realized a cryptographic idea of division of secret splitting.

The distinctive feature of this process is that there is no need to know intermediate harmed sequences to restore the original sequence. Only the final harmed sequence (the final harm after all rounds are over) and all the harms with the rules of harming are required.

Theorem 1.1 *Let M be a sensible text of a length L_0, M_a be a text with the length $L_a < L_0$, that was received from M with the help of "an ideal" archiver, Y_{DT} is a text derived from a text M after executing m rounds of mechanism of harming C_m and its length $L(Y_{\mathrm{DT}}) < L_a$. Then Y_{DT} is a harmed text.*

Proof. Let Y_{DT} be not a harmed text with the length $L_{\mathrm{DT}} < L_a$. Then the transformation $C_m(M)$ is an archiver that generates a sensible text. This contradicts the theorem hypothesis as the transformation of archiving cannot generate a sensible text with the length $L < L_a$. Therefore, the text Y_{DT} is a harmed text. □

Assume that an English source text has a length of L_0 bites. With 8 bit symbols (letters) in accordance with the ASCII table and all punctuation marks and numbers this text contains $B_A = 3,4$ bits of redundant information per symbol. Therefore an ideal archiver can shorten the length of the source text to the value of

$$\frac{8-3,4}{8}L_0 = 0,575L_0,$$

and retain the sence of the source text. Further shortening of the length in a bit dimension will cause a distortion or loss of sense. Let the above described text become η times smaller after every round of a universal harming algorithm. Then the number of rounds m, to destroy the sense can be defined by the following inequality:

$$\frac{L_0}{\eta^m} < 0,575 L_0,$$

that results in:

$$m > \frac{\log 1,739}{\log \eta}.$$

To ensure a high probability of distortion of sense in a harmed text, the number m should be selected based on specific applications and a method of intervention of the harming algorithm into a computing environment. If, for example, someone uses the universal algorithm as an encryption system with observed harmed text and harm, then m should be no less than 16.

It is possible to draw an analogy between ciphertexts and harmed texts.

A ciphertext contains the entire sense of a source text in the alphabet of a ciphertext and changes its appearance without losing sense when a constant encryption algorithm key changes.

All harmed texts and their harms contain the meaning of a source text and change their appearance without losing sense when a rule of harming changes. Here a ciphertext and a set of harmed texts are equivalent to each other.

However, a single harmed text or incomplete set of harmed texts do not contain the sense of a source text.

These statements are based on the definition of sense that requires certain words from a thesaurus that are not available in a harmed text any more because its length was reduced. These statements assume that it is impossible to restore any compressed text to the extent that is more than is allowed by redundant information.

A quantitative measure of a harm effect is the degree of sense distortion being the difference of the entropy in a harmed text and that in a source text at various length segments of the harmed text. The quantity of such segments equals

$$s = \left[\frac{L_0 - L_{\mathrm{DT}}}{L_{\mathrm{DT}}}\right],$$

where L_0 and L_{DT} are lengths of a source text and a harmed text, respectively.

Therefore, the degree of sense distortion in a source text can be found as

$$d = H(Y_{\mathrm{DT}}) - \sum_{i=1}^{s} H(M_i)p_i, \quad \sum_{i=1}^{s} p_i = 1,$$

where M_i is the part of a source text in the ith segment, p_i is probability of M_i, and the length of every M_i equals the length of Y_{DT}.

If a text was generated by an ergodic source, then

$$d = H(Y_{\mathrm{DT}}) - H(M_i).$$

A value d describes the degree of symbol disorder in a harmed text as compared to a source text. With equiprobable distribution of symbols in a harmed text (that represents the maximum harm) the value d has the maximum value

$$d_{max} = \log L_{\mathrm{DT}} - \sum_{i=1}^{s} H(M_i)p_i; \quad \sum_{i=1}^{s} p_i = 1$$

or for an ergodic source of a source text:

$$d_{max} = \log L_{\mathrm{DT}} - H(M_i).$$

The process of generating a harmed text has the following analytical interpretation:

$$\begin{array}{l} Y_{\mathrm{DT}}^{(i)} = D_1^{(i)}(Y_{\mathrm{DT}}^{(i-1)}, K_{\mathrm{D}}^{(i)}) \\ Y_{\mathrm{D}}^{(i)} = D_2^{(i)}(Y_{\mathrm{DT}}^{(i-1)}, K_{\mathrm{D}}^{(i)}) \end{array}, \quad i = 1, 2, ...m,$$

where $Y_{\mathrm{DT}}^{(i)}$ is a harmed text and $Y_{\mathrm{D}}^{(i)}$ is a harm caused in the ith round, $K_{\mathrm{D}}^{(i)}$ is a harm key in the ith round, $Y_{\mathrm{DT}}^{(0)} = M$ is a source text, m is the number of rounds.

The process of restoring a source text has the following interpretation:

$$Y_{\mathrm{DT}}^{(i-1)} = D_i^{-1}(Y_{\mathrm{DT}}^{(i)}, Y_{\mathrm{D}}^{(i)}, K_{\mathrm{D}}^{(i)}), \quad i = m, m-1, \ldots, 1.$$

Fig. 1.1 displays the general scheme of a universal harming algorithm C_m. within one round. The input text $Y_{\mathrm{DT}}^{(i-1)}$ is obtained in the previous round of the algorithm C_m. In the first round it is the source text: $Y_{\mathrm{DT}}^{(0)} = M$.

The scheme implies a special dividing tranformation of the input text $Y_{\mathrm{DT}}^{(i-1)}$ with parameters (key) $K_{\mathrm{D}}^{(i)}$; resulting in two output texts $Y^{(i)}$ and $Y_{\mathrm{D}}^{(i)}$. In general, it is possible that the text $Y^{(i)}$ being a result of the dividing tranformation may be additionally encrypted using the key K_i, resulting at every step in output text $Y_{\mathrm{DT}}^{(i)}$.

Then the universal harming algorithm C_m is described as

$$\begin{array}{l} Y_{\mathrm{DT}} = E_1(M, K) \\ Y_{\mathrm{D}} = E_2(M, K) \\ M = E_{12}^{-1}(Y_{\mathrm{DT}}, Y_{\mathrm{D}}, K) \end{array}, \tag{1.2}$$

where

$$\begin{array}{l} Y_{\mathrm{DT}} = Y_{\mathrm{DT}}^{(m)} \\ K = \phi(K_{\mathrm{D}}^{(1)}, \ldots, K_{\mathrm{D}}^{(m)}, K_1, \ldots, K_m) \\ Y_{\mathrm{DT}} = \psi(Y_{\mathrm{D}}^{(1)}, \ldots, Y_{\mathrm{D}}^{(m)}) \end{array} \tag{1.3}$$

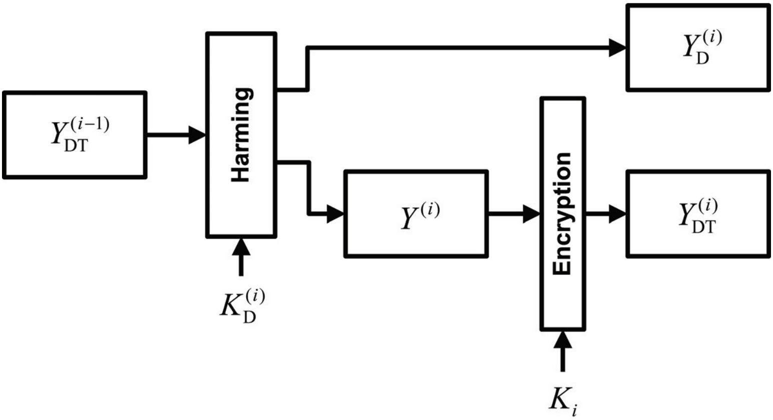

Figure 1.1: *One round of the universal harming algorithm*

and ϕ is appearance for keys $K_{\mathrm{D}}^{(1)}, \ldots, K_{\mathrm{D}}^{(m)}, K_1, \ldots, K_m$, and ψ is some appearance for all harms $Y_{\mathrm{D}}^{(1)}, \ldots, Y_{\mathrm{D}}^{(m)}$, that depends on a specific application.

The described harming algorithm affects all symbols in a text; in our example it affects all bytes. In general, as each symbol of any alphabet is represented by a block of bits, all symbols of the original sequence, and, therefore, the words, will be changed due to symbol deformation beyond their irredundancy and not the procedure of standard substitutions with a fixed length. Thus, the sense of the original sequence that defines these words is destroyed. A harmed sequence becomes shorter not because of compression but of deformation, shortening of a symbol bit length beyond irredundancy which results in losing sense. Evidently, such process can be carried out many times over generated harmed sequences obtaining new harmed sequences and associated harms of the second, third and more levels.

Thus, we come to the idea of multichannel cryptography on the basis of division of a ciphertext into harmed texts.

Such division enables for a cryptanalyst to get the original ciphertext to estimate the unicity distance and to get a sensible text by manipulating the keys when a harmed text or at least one of harms is missing.

Definition 1.4 *A multicannel cryptographic transformation shall be a transformation of source texts resulting in two or more harmed texts that have a cryptographic property of dividing the secret:*

- *any incomplete set of harmed texts does not allow (possibly, except for entire enumeration of missing harmed texts) decryption of the received message;*
- *any complete set of harmed texts transforms the task of decryption into a classical task of a system breaking when a ciphertext is known and with possible greater key space using that of the harming algorithm.*

Destruction of the sense of a source text or a ciphertext in an alphabet of a ciphertext causes additional key opportunities of the encryption process and increase of Shannon's unicity distance. Actually, the very use of this approach results in several ciphertexts, each of them having no sense in an alphabet of a ciphertext, and, respectively, no sensible source text from which it was generated. It is worthy to note the duality of harm text analysis. If a cryptanalyst uses only observed harms without the hypothesis of concealed ciphertexts, then he deals with an ideal encryption system. If he analyses based on concealed ciphertext models, then he deals with an encryption system with a very large unicity distance defined by uncertainty of a harmed text concealed. The need to have all ciphertexts for decryption poses a very difficult problem in front of a cryptanalyst. It is problem of interception while using various data channels and cryptanalysis in the field of harmed cipher texts.

Various methods of harming allow synthesizing cryptographic system models for different applications with different characteristics and properties.

Chapter 2
Usage schemes and modes of multichannel ciphers

2.1 Application of mechanism of harming

A universal mechanism of harming allows creating new applications due to manipulating input and output data.

Let's enumerate variants of data transmission possible at using a two-channel mechanism (1.2).

1. The key K is transmitted via a secure channel, and outputs $Y_{\mathrm{DT}}, Y_{\mathrm{D}}$ are transmitted via an open channel.

This item corresponds to a classical symmetric system.

2. The key K and the output Y_{DT} are transmitted via a secure channel, and the output Y_{D} is transmitted via an open channel.

3. The output Y_{DT} is transmitted via a secure channel, and the key K and the output Y_{D} are transmitted via an open channel.

4. The key K and the output Y_{D} are transmitted via a secure channel, and the output Y_{DT} is transmitted via an open channel.

5. The output Y_{D} is transmitted via a secure channel, and the key K and the output Y_{DT} are transmitted via an open channel.

Moreover, the mechanism of harming can be used for message authentication and in this case the following variants can be considered:

6. The key K and the plaintext M are transmitted via a secure channel, and the outputs $Y_{\mathrm{DT}}, Y_{\mathrm{D}}$ are transmitted via an open channel.

7. The key K and the output Y_{DT} are transmitted via a secure channel, and the output Y_{D} and the plaintext M are transmitted via an open channel.

8. The output Y_{DT} and the plaintext M are transmitted via a secure channel, and the key K and the output Y_{D} are transmitted via an open channel.

9. The key K and the output Y_{D} are transmitted via a secure channel, and the output Y_{DT} and the plaintext M are transmitted via an open channel.

10. The output Y_{D} and the plaintext M are transmitted via a secure channel, and the key K and the output Y_{DT} are transmitted via an open channel.

2.2 Tasks of a two-channel cipher attacker

Let's consider tasks of an adversary cryptanalyst, that arise at using the universal mechanism of harming (1.2). Assume that an attacker knows the type of the mappings E_1, E_2, E_{12}^{-1}. Then, depending on the schemes used for manipulating channels, the following tasks can be considered:

1. An attacker knows the entire ciphertext (Y_{DT}, Y_{D}), he needs to recover:

   a) the corresponding plaintext M;

   b) the key K;

   c) the corresponding plaintext M and the key K.

These tasks are classical at researching symmetric ciphers.

The following tasks arise due to peculiarities of two-channel system use, when an attacker can only observe one of the outputs.

2. An attacker knows the output Y_{DT}, he needs to recover:

   a) the plaintext M;

   b) the key K;

   c) the output Y_{D};

   d) the plaintext M and the key K;

   e) the plaintext M and the output Y_{D};

   f) the key K and the output Y_{D};

   g) the plaintext M, and the key K and the output Y_{D}.

3. An attacker knows the output Y_{DT} and the key K, he needs to recover:

   a) the plaintext M;

   b) the output Y_{D};

   c) the plaintext M and the output Y_{D};

4. An attacker knows the output Y_{DT} and the corresponding plaintext M, he needs to recover:

   a) the key K;

   b) the output Y_{D};

   c) the key K and the output Y_{D};

5. An attacker knows the output Y_{DT} the key K and the plaintext M he needs to recover the output Y_{D}.

6. An attacker knows the output Y_{D}, he needs to recover:

   a) the plaintext M;

   b) the key K;

   c) the output Y_{DT};

   d) the plaintext M and the key K;

   e) the plaintext M and the output Y_{DT};

   f) the key K and the output Y_{DT};

   g) the plaintext M, the key K and the output Y_{DT}.

7. An attacker knows the output Y_{D} and the key K, he needs to recover:

   a) the plaintext M;

   b) the output Y_{DT};

   c) the plaintext M and the output Y_{DT};

8. An attacker observed the output Y_{D} and knows the corresponding plaintext M, he needs to recover:

   a) the key K;

   b) the output Y_{DT};

   c) the key K and the output Y_{DT};

9. An attacker knows the key K, a plaintext M and the output Y_{D}, he needs to recover the output Y_{DT}.

Security of the corresponding schemes of two-channel system application depends on difficulty of solution of the enumerated tasks.

Chapter 3
MV2-transformation

3.1 Substitution transformation for obtaining harmed texts

3.1.1 Mappings with variable length of an image

The substitution stage of block ciphers is composed of different non-linear transformations mapping n-bit values to m-bit ones which are usually called S-boxes. An n-bit to m-bit S-box defines simply a substitution, i.e. to each n-bit input is mapped a corresponding m-bit output value (which has not necessarily the same length than the input). An S-box can be described by a lookup table of 2^n elements of m bits.

Such transformations are called mappings with image strings of a fixed length, or mappings with fixed length outputs.

To build harming transformations, we shall consider mappings of the following kind:

$$f : \{0, 1\}^n \to \bigcup_{i=r}^{m} \{0, 1\}^i.$$

We shall call such mappings as *mappings with image strings of a variable length*, or mappings *with a variable length outputs*. To make it simple, we shall introduce symbols:

$$\mathcal{U}_{rm} = \bigcup_{i=r}^{m} \{0,1\}^i. \tag{3.1}$$

Thus, the set $\{0,1\}^i$ contains 2^i elements, and the set $\mathcal{U}_{rm}$ contains

$$\#\mathcal{U}_{rm} = \sum_{i=r}^{m} 2^i = 2^{m+1} - 2^r$$

various binary strings. Note that if $m < n$, then mappings with variable length outputs can not be injective, because a set of inputs contains 2^n different elements, and a set of outputs contains $2^{m+1} - 2^r < 2^n$.

Mappings with variable length outputs are used in some archivers, for instance. A Huffman code can be an example of such a mapping. [22].

The number of bits in the element $x \in \mathcal{U}_{rm}$ will be denoted by $|x|$. We shall also call the value $|x|$ as the length of the element $x \in \mathcal{U}_{rm}$.

We shall call two binary strings from the set $\mathcal{U}_{rm}$ equal if their lengths and corresponding bits are equal.

3.1.2 Definition of the MV2-transformation

To build harming transformations we shall consider substitution transformations that map binary strings of the length n bits into variable length strings, such that the maximum length is *smaller*, than n. As the number of elements in the domain of such a transformation is more than that in the range, then such transformations are not injective mappings.

But, to restore a plaintext from a ciphertext the transformation of a plaintext needs to be an invertible mapping.

Let r and n be integers such that $0 < r < n$. Consider the mappings

$$c : \{0,1\}^n \to \mathfrak{U}_{r\ n-1},$$

having the following properties:

1) c is surjective: for each element $y \in \mathfrak{U}_{r\ n-1}$ there is at least one element $x \in \{0,1\}^n$, such that $c(x) = y$;

2) the restriction of c to $\mathfrak{U}_{r+1\ n-1}$ is bijection;

3) for each element $y \in \{0,1\}^r$ has only two preimages.

For every mapping c we shall define an integer function

$$f : \{0,1\}^n \to \{1,\ldots,n-r+1\}$$

connected with it in the following way:

1) $f(x) = n - |c(x)|$, if $|c(x)| > r$;

2) if $x_1, x_2 \in \{0,1\}^n$, $x_1 \neq x_2$ and $c(x_1) = c(x_2) \in \{0,1\}^r$, then $f(x_1) \neq f(x_2)$ and $f(x_1), f(x_2) \in \{n-r, n-r+1\}$.

It is clear, that for a fixed mapping c there are 2^{2^r} various functions f. Every pair (c, f) induce an injective mapping of the kind:

$$\begin{aligned} T : \{0,1\}^n \ \to & \bigcup_{i=1}^{n-r-1} \Big\{\{0,1\}^{n-i} \times \{i\}\Big\} \bigcup \\ & \bigcup \Big\{\{0,1\}^r \times \{n-r, n-r+1\}\Big\}, \end{aligned} \tag{3.2}$$

which transforms a binary string of a fixed length into the pair: (a binary string of a variable length, a number). We shall denote a set of such mappings as $\mathcal{F}_n^r$. We shall further call these mappings as *MV2-mappings* (MV2-transformations).

Output values $f(x)$ of a transformation $T = (c, f)$ can be encoded by a binary code (BC) as shown in the table 3.1. In this table 0^i indicates a bit string of i zeros.

Table 3.1: *Value encoding f*

$f(x)$	1	2	3	...	$n-r$	$n-r+1$
BC	1	01	$0^2 1$	...	$0^{n-r-1}1$	0^{n-r}

This cod coincides with the Huffman code of the alphabet $\{1, \ldots, n-r+1\}$ with distribution P, such that $P(Y = i) = 2^{-i}$ for $1 \leq i \leq n-r$ and $P(Y = n-r+1) = 2^{-r}$.

Let $x \in \{0,1\}^n$. We shall call the image $c(x) \in \mathcal{U}_{r\ n-1}$ as *a remainder*, and the cod of $f(x)$ as *a flag*.

It's obvious that any mapping from $\mathcal{F}_n^r$ can be set with the help of the table, in the first column of which there is a permutation $(s_1, \ldots, s_{2^n})$ of n-bit binary strings, and in the right part there are images consisting of "remainer" and "flag" parts, as it is shown in the table 3.2. In this table 0^i and 1^i denote binary strings of i zeros and ones correspondingly.

If in the table 3.2 we fix the right columns, then different permutations in first column will correspond to different MV2-transformations. Thus, there is a bijection between the set of permutations $\big\{(s_1, \ldots, s_{2^n})\big\}$ and the set $\mathcal{F}_n^r$. Therefore:

$$\#\mathcal{F}_n^r = \#\Big\{\,(\,c, f\,)\,\Big\} = 2^n!.$$

In the table 3.3 there is an example of MV2-transformation with parameters $n = 4$ and $r = 2$.

Note, that any flag mapping f splits the set $\{0,1\}^n$ into $n-r+1$ noncrossing subsets $\mathcal{X}_i$ such that $\forall x \in \mathcal{X}_i :\ f(x) = i$ and in every set $\mathcal{X}_i$, for $n-r \leq i \leq n-1$ exactly 2^{n-i} elements are contained and the set $\mathcal{X}_{n-r+1}$ contains 2^r elements.

Table 3.2: *Presentation of a substituation transformation*

a symbol	a remainder length	a remainder	a flag
s_1	r	0^r	$0^{n-r-1}1$
s_2	r	$0^{r-1}1$	$0^{n-r-1}1$
$\dots$	$\dots$	$\dots$	$\dots$
s_{2^r+1}	$r+1$	0^{r+1}	$0^{n-r-2}1$
$\dots$	$\dots$	$\dots$	$\dots$
$s_{2^{n-1}-2^r}$	$n-2$	1^{n-2}	01
$s_{2^{n-1}-2^r+1}$	$n-1$	0^{n-1}	1
$\dots$	$\dots$	$\dots$	$\dots$
$s_{2^n-2^r}$	$n-1$	1^{n-1}	1
$s_{2^n-2^r+1}$	r	0^r	0^{n-r}
$\dots$	$\dots$	$\dots$	$\dots$
s_{2^n}	r	1^r	0^{n-r}

Thus, there are as many various flag mappings as methods of the set splitting $\{0,1\}^n$ into $n-r+1$ noncrossing subsets $\mathfrak{X}_i$.

The number of ways to pick elements for $\mathfrak{X}_1$ is equal to $\binom{2^n}{2^{n-1}}$, then the number of ways to pick elements for $\mathfrak{X}_2$ is equal to $\binom{2^{n-1}}{2^{n-2}}$ and so on.

Let $\mathfrak{F}$ denote the set of all flag mappings f.

The set $\mathfrak{F}$ contains

$$\#\mathfrak{F} = \prod_{i=r}^{n-1} \binom{2^{i+1}}{2^i} = \frac{2^n!}{\prod\limits_{i=r}^{n-1} 2^i!} \tag{3.3}$$

different mappings f.

Table 3.3: *Example of an MV2-transformation for $n = 4$ and $r = 2$*

$\mathbf{x}$	0000	0001	0010	0011
$\mathbf{c}(\mathbf{x})$	111	110	10	01
$\mathbf{f}(\mathbf{x})$ (BC)	1(1)	1(1)	2(01)	3(00)

$\mathbf{x}$	0100	0101	0110	0111
$\mathbf{c}(\mathbf{x})$	10	11	010	11
$\mathbf{f}(\mathbf{x})$ (BC)	3(00)	2(01)	1(1)	3(00)

$\mathbf{x}$	1000	1001	1010	1011
$\mathbf{c}(\mathbf{x})$	00	001	100	00
$\mathbf{f}(\mathbf{x})$ (BC)	2(01)	1(1)	1(1)	3(00)

$\mathbf{x}$	1100	1101	1110	1111
$\mathbf{c}(\mathbf{x})$	101	000	011	01
$\mathbf{f}(\mathbf{x})$ (BC)	1(1)	1(1)	1(1)	2(01)

Similarly, the number of different remainder mappings is

$$\frac{2^n!}{2^{n-1}!} \cdot \frac{2^{n-1}!}{2^{n-2}!} \cdot \ldots \cdot \frac{2^{r+1}!}{2^r!} \cdot 2^r! = 2^n!$$

i.e. it is equal to the number of different MV2-mappings.

Let $y \in \mathcal{U}_{r\ n-1}$ be a binary string, and $i \in \{1, 2, \ldots, |y|$. We shall denote by $y^{(i)}$ a binary string, which is obtained from the string y by the inversion of the i-th bit (For instance, $y = 01001010$, then $y^{(3)} = 01101010$).

Lemma 3.1 *Let $T = (c, f) \in \mathcal{F}_n^r$ be an MV2-transformation. Then for any $y \in \mathcal{U}_{r\ n-1}$ and for any $1 \leq i \leq |y|$ the following is carried out*

$$\#\Big\{x \in \{0,1\}^n : c(x) = y\Big\} = \#\Big\{x \in \{0,1\}^n : c(x) = y^{(i)}\Big\}.$$

3.1.3 Information estimations for an MV2-transformation

During harming text symbols are replaced by binary strings of various length. In the previous paragraph we defined a family of transformations, which can be used to harm arbitrary texts. The domain of an MV2-transformation is a set $\{0,1\}^n$, it can be considered as an alphabet of plaintexts. Let a probability distribution for the letters in the alphabet is given. Then the input of the MV2-transformation is a random vector and the outputs are random strings. Besides, as outputs of the MV2-transformation are binary strings of variable length, the output lengths can also be considered as random variables.

Let X be a discrete random variable with possible values $x_i \in \{0,1\}^n$ which have probabilities p_i, $i = 1, 2, \dots 2^n$, and let $T = (c, f)$ be an MV2-transformation, where $c : \{0,1\}^n \to \mathfrak{U}_{r\ n-1}$ is a remainder mapping with images of variable length, and $f : \{0,1\}^n \to \{1, 2, \dots, n-r+1\}$ is a flag mapping. The random pair $Y_{\mathrm{DT}}, Y_{\mathrm{D}}$, where $Y_{\mathrm{DT}} = c(X)$ and $Y_{\mathrm{D}} = f(X)$ is an image of the random element X. The random variables Y_{DT} and Y_{D} are random strings, and their lengths $|Y_{\mathrm{DT}}|$ and $|Y_{\mathrm{D}}|$ are random values.

For the joint entropy of random elements X, Y_{DT} and Y_{D} the following identities are satisfied [17]:

$$\begin{aligned} H(XY_{\mathrm{DT}}Y_{\mathrm{D}}) &= H(X) + H(Y_{\mathrm{DT}}Y_{\mathrm{D}}|X), \\ H(XY_{\mathrm{DT}}Y_{\mathrm{D}}) &= H(Y_{\mathrm{DT}}Y_{\mathrm{D}}) + H(X|Y_{\mathrm{DT}}Y_{\mathrm{D}}). \end{aligned}$$

For a fixed transformation $T \in \mathcal{F}_n^r$ the pair $(Y_{\mathrm{DT}}, Y_{\mathrm{D}}) = T(X)$ is calculated X, therefore

$$H(Y_{\mathrm{DT}}Y_{\mathrm{D}}|X) = H(X|Y_{\mathrm{DT}}Y_{\mathrm{D}}) = 0.$$

Consequently, for the joint entropy Y_{DT} and Y_{D} the following is satisfied:

$$H(Y_{\mathrm{DT}}Y_{\mathrm{D}}) = H(X). \tag{3.4}$$

On the other hand

$$H(Y_{\mathrm{DT}}Y_{\mathrm{D}}) = H(Y_{\mathrm{DT}}) + H(Y_{\mathrm{D}}|Y_{\mathrm{DT}}) = H(Y_{\mathrm{D}}) + H(Y_{\mathrm{DT}}|Y_{\mathrm{D}}).$$

From that we have the following, using (3.4):

$$H(Y_{\mathrm{DT}}|Y_{\mathrm{D}}) = H(X) - H(Y_{\mathrm{D}}), \tag{3.5}$$

$$H(Y_{\mathrm{D}}|Y_{\mathrm{DT}}) = H(X) - H(Y_{\mathrm{DT}}). \tag{3.6}$$

By definition and (3.5) the mutual information between Y_{DT} and Y_{D} is

$$I(Y_{\mathrm{DT}}, Y_{\mathrm{D}}) = H(Y_{\mathrm{DT}}) + H(Y_{\mathrm{D}}) - H(X). \tag{3.7}$$

We shall consider informational dependencies between the input and outputs.

If T is fixed, the output Y_{DT} is completely determine by the input X, therefore $H(Y_{\mathrm{DT}}|X) = 0$.

Then the mutual information between X and Y_{DT} equals

$$I(X, Y_{\mathrm{DT}}) = H(Y_{\mathrm{DT}}). \tag{3.8}$$

For the joint entropy of X and Y_{DT} the following is carried out:

$$H(XY_{\mathrm{DT}}) = H(X) + H(Y_{\mathrm{DT}}|X) = H(Y_{\mathrm{DT}}) + H(X|Y_{\mathrm{DT}}).$$

From which we have

$$H(X) = H(Y_{\mathrm{DT}}) + H(X|Y_{\mathrm{DT}}). \tag{3.9}$$

Similarly, as Y_{D} is the part of the image X and the transformation T is fixed, then $H(Y_{\mathrm{D}}|X) = 0$, therefore, for the joint entropy and the mutual information between the random variables X and Y_{D} the following is carried out:

$$H(X) = H(Y_{\mathrm{D}}) + H(X|Y_{\mathrm{D}}), \tag{3.10}$$

$$I(X, Y_{\mathrm{D}}) = H(Y_{\mathrm{D}}). \tag{3.11}$$

If $T = (c, f)$ is an MV2-transformation with the parameters r and n, then the flag mappings f split the domain into noncrossing subsets $\mathcal{X}_1, \dots \mathcal{X}_{n-r+1} \subset \{0,1\}^n$, such, that for all $x \in \mathcal{X}_i$ the following is carried out: $f(x) = i$.

Let the inputs x_i is numbered such that for $i = 1, \dots, 2^{n-1}$ the images $c(x_i) \in \{0,1\}^{n-1}$ and $f(x_i) = 1$; for $i = 2^{n-1} + 1, \dots, 2^{n-1} + 2^{n-2}$ the images $c(x_i) \in \{0,1\}^{n-2}$, and $f(x_i) = 2$; $\dots$, for $i = 2^n - 2^{r+2} + 1, \dots, 2^n - 2^{r+1}$ images $c(x_i) \in \{0,1\}^{r+1}$ and $f(x_i) = n - r - 1$; for $i = 2^n - 2^{r+1} + 1, \dots, 2^n - 2^r$ images $c(x_i) \in \{0,1\}^r$ and $f(x_i) = n - r$; and, finally, for $i = 2^n - 2^r + 1, \dots, 2^n$ images $c(x_i) \in \{0,1\}^r$ and $f(x_i) = n - r + 1$.

Then for the remainder entropy $H(Y_{\mathrm{DT}})$ we have

$$H(Y_{\mathrm{DT}}) = -\sum_{i=1}^{2^n - 2^{r+1}} p_i \log p_i - \sum_{i=2^n-2^{r+1}+1}^{2^n - 2^r} (p_i + p_i') \log(p_i + p_i'),$$

where $p_i' = p_{i+2^r}$ for $i = 2^n - 2^{r+1} + 1, \dots, 2^n - 2^r$.

Therefore the random elements X and Y_{DT} have different distributions in the general case.

Difference of input and output entropies is

$$H(X) - H(Y_{\mathrm{DT}}) = \sum_{i=2^n-2^{r+1}+1}^{2^n - 2^r} \big(p_i \log(1 + \frac{p_i'}{p_i}) + p_i' \log(1 + \frac{p_i}{p_i'})\big),$$

and can be estimated as

$$0 \le H(X) - H(Y_{\mathrm{DT}}) \le \sum_{i=2^n-2^{r+1}+1}^{2^n} p_i \log(1 + \frac{1}{p_i}).$$

Denote by P_r the probability event $f(X) = r$, then

$$P_r = P(c(X) \in \{0,1\}^r) = \sum_{i=2^n-2^{r+1}+1}^{2^n} p_i$$

and

$$\sum_{i=2^n-2^{r+1}+1}^{2^n} p_i \log(1 + \frac{1}{p_i}) = P_r \sum_{i=2^n-2^{r+1}+1}^{2^n} \frac{p_i}{P_r} \log(1 + \frac{1}{p_i}).$$

As the function $x \log(1 + \frac{1}{x})$ is convex, then, due to the Jensen inequality we have

$$\sum_{i=2^n-2^{r+1}+1}^{2^n} p_i \log(1 + \frac{1}{p_i}) \leq \Big(\sum_{i=2^n-2^{r+1}+1}^{2^n} p_i \Big) \log(1 + \frac{P_r}{\sum p_i}).$$

Thus, the following estimation is true:

$$0 \leq H(X) - H(Y_{\mathrm{DT}}) \leq P_r. \tag{3.12}$$

At that, the equality can be reached only if all inputs x_i with r-bit images have the same probability $p = p_i$.

For equiprobable inputs all probabilities $p_i = 1/2^n$, $i = 1, \ldots, 2^n$, therefore, we have the following from (3.12):

$$H(X) - H(Y_{\mathrm{DT}}) = n - H(Y_{\mathrm{DT}}) = 2^{r+1-n}. \tag{3.13}$$

Similarly, for flag entropy $H(Y_{\mathrm{D}})$ we have:

$$\begin{aligned} H(Y_{\mathrm{D}}) = \; & \Big(\sum_{i=1}^{2^{n-1}} p_i\Big) \log \frac{1}{\sum\limits_{i=1}^{2^{n-1}} p_i} + \ldots + \\ & + \Big(\sum_{i=2^n-2^{r+1}+1}^{2^n-2^r} p_i\Big) \log \frac{1}{\sum\limits_{i=2^n-2^{r+1}+1}^{2^n-2^r} p_i} + \\ & + \Big(\sum_{i=2^n-2^r+1}^{2^n} p_i\Big) \log \frac{1}{\sum\limits_{i=2^n-2^r+1}^{2^n} p_i}. \end{aligned}$$

Let

$$P_k = \sum_{\{x:f(x)=k\}} P(X = x) \tag{3.14}$$

be the probability that a flag image will possess the value $k = 1, \ldots, n - r + 1$. Then

$$\begin{aligned} H(X) - H(Y_{\mathrm{D}}) \; & = \sum_{i=1}^{2^{n-1}} p_i \log \tfrac{P_1}{p_i} + \ldots + \\ & + \sum_{i=2^n-2^{r+1}+1}^{2^n-2^r} p_i \log \tfrac{P_{n-r}}{p_i} + \\ & + \sum_{i=2^n-2^r+1}^{2^n} p_i \log \tfrac{P_{n-r+1}}{p_i}. \end{aligned}$$

From here, using again Jensen inequality we get:

$$H(X) - H(Y_{\mathrm{D}}) \leq \sum_{k=1}^{n-r} (n-k) \cdot P_k + r \cdot P_{n-r+1}.$$

taking into account that $\sum\limits_{k=1}^{n-r+1} P_k = 1$, we have:

$$H(X) - H(Y_{\mathrm{D}}) \leq n + P_{n-r+k} - \sum_{k=1}^{n-r+1} k \cdot P_k, \tag{3.15}$$

In the inequality (3.15) equality is reached only if for every $k = 1, \dots, n - r + 1$ probabilities of all the preimages $x \in \mathcal{X}_k$ are equal to $P(X = x : f(x) = k) = \frac{P_k}{2^{|f(x)|}}$ for all $x \in \mathcal{X}_k$, where $|f(x)|$ is a number of bits in the representation of the value $f(x)$ in form of a bit string (see table 3.1).

Whatever probability distribution of the random element X, is, for the entropy of the random elements Y_{DT} and Y_{D} the following inequalities are true:

$$H(Y_{\mathrm{DT}}) \leq \log(2^n - 2^r) \tag{3.16}$$

$$H(Y_{\mathrm{D}}) \leq \log(n - r + 1) \tag{3.17}$$

3.1.4 Information estimations for outputs at uniform input distribution

In this section we consider an important special case when the inputs x for an MV2-transformation $T = (c, f)$, are uniformly distributed, i.e. probabilities of any symbol $x \in \{0,1\}^n$ coincide and equal $\dfrac{1}{2^n}$.

In this case the remainder probability $P(Y_{\mathrm{DT}} = y)$ and the flag probability $P(Y_{\mathrm{D}} = k)$ are equal:

$$P(Y_{\mathrm{DT}} = y) = \begin{cases} 2^{-n}, & \text{if } y \in \bigcup\limits_{i=r+1}^{n-1} \{0,1\}^i \\ 2^{1-n}, & \text{if } y \in \{0,1\}^r \end{cases}; \tag{3.18}$$

$$P(Y_{\mathrm{D}} = k) = \begin{cases} 2^{-k}, & \text{if } 1 \leq k \leq n - r \\ 2^{r-n}, & \text{if } k = n - r + 1 \end{cases}. \tag{3.19}$$

If $T = (c, f)$ is a fixed MV2-transformation and the random element $X \in \{0,1\}^n$ has a uniform distribution, then the entropy $H(X) = n$. Therefore the following eqality follows the form(3.8), (3.12), (3.18)

$$I(X; Y_{\mathrm{DT}}) = H(Y_{\mathrm{DT}}) = n - 2^{r+1-n}. \tag{3.20}$$

Similarly, from expressions (3.11), (3.15), (3.19) and identity

$$\sum_{k=1}^{m} k \cdot 2^k = (m-1)2^{m+1} + 2$$

we have

$$I(X; Y_{\mathrm{D}}) = H(Y_{\mathrm{D}}) = 2 - 2^{r+1-n}. \tag{3.21}$$

From (3.5), (3.10) and (3.21) we have:

$$H(X|Y_{\mathrm{D}}) = H(Y_{\mathrm{DT}}|Y_{\mathrm{D}}) = n - 2 + 2^{r+1-n}. \tag{3.22}$$

And from (3.9) and (3.20) we shall get:

$$H(X|Y_{\mathrm{DT}}) = 2^{r+1-n}, \tag{3.23}$$

Accordingly, it follows from (3.7), (3.20) and (3.21), that:

$$I(Y_{\mathrm{DT}}; Y_{\mathrm{D}}) = 2 - 2^{r+2-n}. \tag{3.24}$$

3.1.5 Estimations of output lengths for an MV2-transformation

Let $T = (c, f)$ be an MV2-transformation. If we consider an input as a random element $X \in \{0,1\}^n$, then the remainder output $Y_{\mathrm{DT}} = c(X)$ is a random element which possesses values from a set of variable length binary strings. Correspondingly, the output length of the remainder $|Y_{\mathrm{DT}}|$ is a numeric random value. As it was mentioned above, the flag output can be encoded by a binary code (see table 3.1), in this case the output $Y_{\mathrm{D}} = f(X)$ is a random element possessing values from a set of variable length strings, and the output length of the flags $|Y_{\mathrm{D}}|$ is also a numeric random value.

Using the designation (3.14), in the general case mathematical expectations of output lengths of the remainder and flags can be represented by the expression:

$$\mathbf{E}(|Y_{\mathrm{DT}}|) = r \cdot P_{n-r+1} + \sum_{k=1}^{n-r} (n-k) \cdot P_k; \tag{3.25}$$

$$\mathbf{E}(|Y_{\mathrm{D}}|) = (n-r) \cdot P_{n-r+1} + \sum_{k=1}^{n-r} k \cdot P_k. \tag{3.26}$$

In case of uniform distribution of inputs for mathematical expectations and random value dispersions $|Y_{\mathrm{DT}}|$ and $|Y_{\mathrm{D}}|$ the following is true:

Claim 3.1 *If $T = (c, f) \in \mathcal{F}_n^r$, is fixed and inputs have uniform distribution then for the mathematical expectations $\mathbf{E}(|Y_{\mathrm{DT}}|)$, $\mathbf{E}(|Y_{\mathrm{D}}|)$ and dispersions $D(|Y_{\mathrm{DT}}|)$, $D(|Y_{\mathrm{D}}|)$ of output lengths the following equalities are true:*

$$\mathbf{E}(|Y_{\mathrm{DT}}|) = n - 2 + 2^{r+1-n}, \tag{3.27}$$

$$\mathbf{E}(|Y_{\mathrm{D}}|) = 2 - 2^{r+1-n}, \tag{3.28}$$

$$\mathbf{D}(|Y_{\mathrm{DT}}|) = 2 - \frac{2n-2r-1}{2^{n-r-1}} - \frac{1}{4^{n-r-1}}, \tag{3.29}$$

$$\mathbf{D}(|Y_{\mathrm{D}}|) = 2 + \frac{(n-r)^2 + 2(n-r) - 1}{2^{n-r-1}} - \frac{1}{4^{n-r-1}}. \tag{3.30}$$

Proof. The proof of the statement 3.1 is obtained with the help of a direct computation.

In fact, the probability $P(Y_{\mathrm{DT}} \in \{0,1\}^k)$ is determined from (3.18), and $P(Y_{\mathrm{D}} = k)$ from (3.19). To prove the equality

(3.27) and (3.28), it's enough to substitute the corresponding probability values.

To prove the equality (3.29) and (3.30), it's enough to substitute the corresponding probability values for dispersion definition and use the equality

$$\sum_{k=1}^{m} k^2 \cdot 2^{-k} = 6 - (m^2 + 4m + 6) \cdot 2^{-m}.$$

□.

3.2 Using MV2-transformations for coding texts

Modern computer systems operate with machine words which are binary strings of a fixed length, that is 8, 16, 32 and so on bits, as a rule. The minimal discrete is a bit, that is a bit possessing the value 0 or 1. The minimal addressable discrete is usually a byte that consist of eight bits. Assume that a plaintext consists of sequential symbols, each of them is being chosen from a finite alphabet. In computer systems any text can be considered as a concatenation of bytes. The MV2-transformations that are defined on a binary string set of a fixed length n allow extending the domain for a set binary strings having lengths divisible by n.

Let a plaintext alphabet contain 2^n letters and symbols coding binary strings $\{0,1\}^n$. Then (the cod of) a plaintext M is a concatenation of elements from $x_i \in \{0,1\}^n$:

$$M = x_1 \| x_2 \| \ldots \| x_L,$$

where L denotes the number of symbols in the text.

Definition 3.1 *Let $T = (c, f) \in \mathcal{F}_n^r$ be an MV2-transformation and $M = x_1 \| x_2 \| \ldots \| x_L$ be a text representing a concatenation of L symbols $x_i \in \{0, 1\}^n$. We shall call the result of using the transformation T to the text M the pair $\Big(c(M), f(M)\Big)$ of binary strings, where:*

$$\begin{aligned} c(M) &= c(x_1) \| c(x_2) \| \ldots \| c(x_L), \\ f(M) &= f(x_1) \| f(x_2) \| \ldots \| f(x_L). \end{aligned} \tag{3.31}$$

As before, the image $c(M)$ of the text M is called *a remainder*, and $f(M)$ is called *flags*. Thus, the remainder of the texts M is a concatenation of image remainders $c(x_i)$, and the flags of the text M is a concatenation of image flags $f(x_i)$ (see (3.31)).

Example 3.1 *Let an MV2-transformation (c, f) is defined by the table 3.3.*

Let the plaintext

$$M = 0010 \| 0100 \| 0010 \| 1001 \| 0110 \| 0110 \| 0110 \| 1000$$

is given as the input. The result of the transformation is presented in the table:

M	0010	0100	0010	1001	0110	0110	0110	1000
$c(M)$	10	10	10	001	010	010	010	00
$f(M)$	01	00	01	1	1	1	1	01

We get the remainder:
$c(M) = 1010 \| 1000 \| 1010 \| 0100 \| 1000$
and the flags:
$f(M) = 0100 \| 0111 \| 1101.$

3.2.1 The number of remainder and flags preimages

Let $T = (c, f)$ be an MV2-transformation. Let the set $\{M\}$, contains all texts consisting of L symbols. The length of $c(M)$ is no more than the length of M. In the general case there are several preimages of $c(M)$.

Let $Y_{\mathrm{DT}} = c(M)$ be the remainder of the text M, and $Y_{\mathrm{D}} = f(M)$ be the flags of the text M. We shall further call the plaintext M as an input of the MV2-transformation, and an obtained remainder and flags of the text M – as outputs.

Let $L = |M|$ and $l = |Y_{\mathrm{DT}}|$ then

$$L \cdot r \ \leq \ l \ \leq \ L \cdot (n-1)$$

and the number of preimages of the remainder Y_{DT} depends on its length. In this section we shall find this dependence.

Let $\widetilde{K}_{l\ s}(r, n)$ denote the number of various integer solutions of equation:

$$z_1 + \ldots + z_s = l, \quad r \leq z_i < n, \ i = 1, \ldots, s. \qquad (3.32)$$

The number of solutions of the equation (3.32) equals:

$$\widetilde{K}_{l\ s}(r, n) = \sum_{k=0}^{m} (-1)^k \binom{s}{k} \binom{l - rs - (n-r)k + s - 1}{s - 1},$$

where the limit superior $m = \min\left\{s, \dfrac{l - sr}{n - r}\right\}$.

Let $N_l^{(c)}$ be the number of various preimages for an image of the length l. Then

$$N_l^{(c)} = 2^L \cdot \delta(l - rL) + \sum_{i=0}^{L-1} 2^i \binom{L}{i} \widetilde{K}_{l-r\cdot i\ L-i}(r+1, n), \quad (3.33)$$

where $\delta(x) = \begin{cases} 0, & \text{if } x \neq 0 \\ 1, & \text{if } x = 0 \end{cases}$

Note, that for $N_l^{(c)}$ the following estimation is true:

$$N_l^{(c)} \leq 2^{nL-l}, \quad rL \leq l \leq (n-1)L. \tag{3.34}$$

As the possible number of texts, consisting of the concatenation of L n-bits strings, equals 2^{nL}, the following equality is satisfied:

$$\sum_{l=rL}^{(n-1)L} 2^l \cdot N_l^{(c)} = 2^{nL}. \tag{3.35}$$

In Fig. 3.1 there is a chart of the dependence of the number of preimages on an image length for 7-byte inputs and the MV2-transformation with the parameters $r = 3$ and $n = 8$.

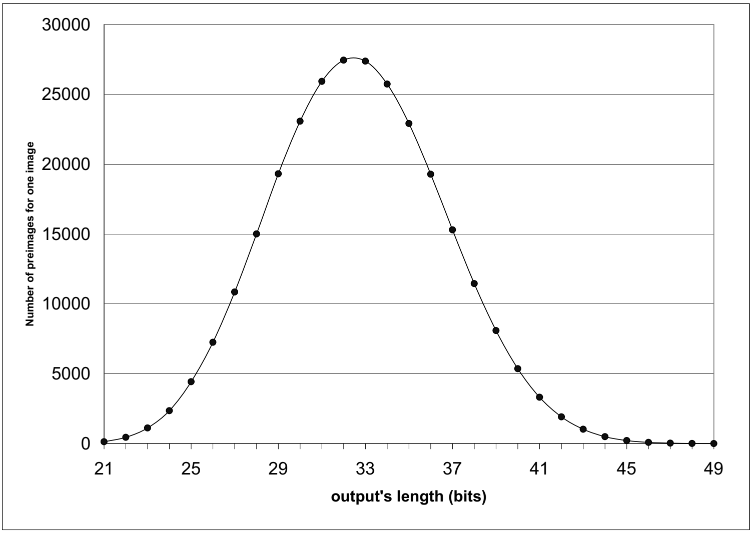

Figure 3.1: *The number of preimages for reminder outputs of different length when $L = 7,\ n = 8,\ r = 3$*

We shall estimate the number of possible preimages for flag outputs. The flag output

$$Y_{\mathrm{D}} = f(M) = f(x_1)\| \dots \|f(x_L)$$

can possess $(n-r+1)^L$ various values. Let k_j letters x_i of M have the flag output $f(x_i) = j$, where $j = 1, \dots, n-r+1$. Then

$$\sum_{j=1}^{n-r+1} k_j = L, \qquad 0 \le k_j \le L \tag{3.36}$$

and the number of flag preimages $N^{(f)}$ equal

$$N^{(f)} = 2^{k_{n-r+1}} \cdot \prod_{j=1}^{n-r+1} 2^{(n-j)\cdot k_j}. \tag{3.37}$$

Due to (3.3) there are $\prod\limits_{i=1}^{n-1} 2^i!$ mappings $T = (c, f) \in \mathcal{F}_n^r$, giving the same flag images in all texts.

3.2.2 Evaluations of output lengths of the remainder and the flags of an MV2-transformation

Output lengths are of great importance for analysis of MV2-transformations. It's impossible to evaluate estimated values of output lengths in the general case. Therefore, we won't go beyond a special case when an input text M is random and uniformly chosen from the set $\{0,1\}^{nL}$. There is an example of the probability distribution of remainder lengths for 7-byte equiprobable inputs in case of an arbitrary MV2-transformation with the parameters $r = 3$ and $n = 8$ in Fig. 3.2.

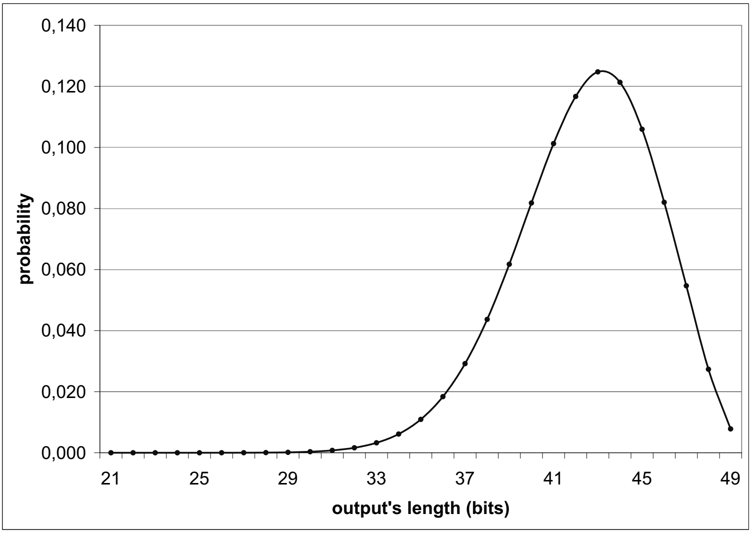

Figure 3.2: *A chart of probability distribution of remainder output lengths at a uniform distribution of 7-byte inputs for an arbitrary fixed MV2-transformation with the parameters* $n = 8,\ r = 3$.

Let $T = (c, f)$ be an MV2-transformation. Let $M = x_1\| \ldots \|x_L$ be a plaintext consisting of L n-bit strings $x_i \in \{0,1\}^n$, and let $Y_{\mathrm{DT}} = c(M),\ Y_{\mathrm{D}} = f(M)$ be images of the text M at the transformation T.

Then mathematical expectation of the remainder output length is:

$$\mathbf{E}(|Y_{\mathrm{DT}}|) = 2^{-n \cdot L} \sum_{i=r \cdot L}^{(n-1)\cdot L} i \cdot N_i^{(c)} \cdot 2^i,$$

where $N_i^{(c)}$ is the number of various preimages for the remainder output length i.

On the other hand, as the text M is being randomly and uniformly chosen from the set $\{0,1\}^{nL}$, the remainder output length can be considered as the sum of integer independent random values $|c(X_i)|$, $X_i \in \{0,1\}^n$. Due to (3.27) for mathematical expectation of the remainder length at uniformly distributed inputs we have

$$\mathbf{E}(|Y_{\mathrm{DT}}|) = \big(n - 2 + 2^{r+1-n}\big) \cdot L.$$

Thus the following equality is satisfied:

$$2^{-n \cdot L} \sum_{i=r\cdot L}^{(n-1)\cdot L} i \cdot N_i^{(c)} \cdot 2^i = \big(n - 2 + 2^{r+1-n}\big) \cdot L. \tag{3.38}$$

Similarly, the output length of the flags $|Y_D|$ can be considered as the sum of independent integer random values $|f(X_i)|$, $X_i \in \{0,1\}^n$. Then, due to (3.28) for mathematical expectation of flags output length we have

$$\mathbf{E}(|Y_D|) = \big(2 - 2^{r+1-n}\big) \cdot L.$$

Thus, the following statement is true

Claim 3.2 *Let a text $M = x_1 \| \ldots \| x_L$, consist of L symbols x_i, being independently of one another, randomly and uniformly chosen from $\{0,1\}^n$, and let some $T = \Big(c, f\Big) \in \mathfrak{F}_n^r$ be fixed. Let $Y_{\mathrm{DT}} = c(M)$ be the remainder and $Y_{\mathrm{D}} = f(M)$ be the flags obtained in the result of using the transformation T for the text M, and let $|Y_{\mathrm{DT}}|$, $|Y_{\mathrm{D}}|$ be their lengths. Then for the mathematical expectation of the remainder length $\mathbf{E}\Big(|Y_{\mathrm{DT}}|\Big)$ and the mathematical expectation of the flag length $\mathbf{E}\Big(|Y_D|\Big)$ the following is satisfied*

$$\mathbf{E}(\Big(|Y_{\mathrm{DT}}|\Big) = \Big(n - 2 + 2^{r-n+1}\Big) \cdot L; \tag{3.39}$$

$$\mathbf{E}(\Big(|Y_{\mathrm{D}}|\Big) = \Big(2 - 2^{r-n+1}\Big) \cdot L; \tag{3.40}$$

The statement 3.2 allows finding coefficients that show estimated decrease of the output length of the MV2-transformation relative to the length of a plaintext.

Definition 3.2 *We shall call the number*

$$\mathrm{K_c} = 1 - \frac{2 - 2^{r-n+1}}{n}. \tag{3.41}$$

as the compression ratio of the remainder at the transformation $T = \Big(c, f\Big) \in \mathfrak{F}_n^r$.

Definition 3.3 *We shall call the number*

$$\mathrm{K_f} = \frac{2 - 2^{r-n+1}}{n}. \tag{3.42}$$

as the compression ratio of flags at the transformation $T = \Big(c, f\Big) \in \mathfrak{F}_n^r$.

It's evident, that

$$\mathrm{K_c} = 1 - \mathrm{K_f}. \tag{3.43}$$

To evaluate probabilities of output lengths rejection from the average value the following is satisfied

Claim 3.3 *Let the text $M = x_1 \| \ldots \| x_L$, consist of L symbols x_i, being independently of one another, random and uniformly chosen from $\{0,1\}^n$ and some $T = \Big(c, f\Big) \in \mathfrak{F}_n^r$ be fixed. Then for probabilities of length rejection of the obtained remainder $|Y_{\mathrm{DT}}| = |c(M)|$ from $\mathbf{E}(|Y_{\mathrm{DT}}|)$ and the one of the obtained flags $|Y_{\mathrm{D}}| = |f(M)|$ from $\mathbf{E}(|Y_{\mathrm{D}}|)$ the following is satisfied*

$$P\left(\Big||Y_{\mathrm{DT}} - \mathbf{E}(|Y_{\mathrm{DT}}|)\Big| < \sigma_c \cdot L\right) > 1 - \frac{1}{L^2}, \tag{3.44}$$

$$P\left(\Big||Y_{\mathrm{D}}| - \mathbf{E}(|Y_{\mathrm{D}}|)\Big| < \sigma_f \cdot L\right) > 1 - \frac{1}{L^2}, \tag{3.45}$$

where

$$\sigma_c = \sqrt{2 - (2n - 2r - 1) \cdot 2^{r-n+1} - 4^{r-n+1}},$$

$$\sigma_f = \sqrt{2 + ((n-r)^2 + 2(n-r) - 1) \cdot 2^{r-n+1} - 4^{r-n+1}}.$$

The proof of the statement 3.3 follows from the statements 3.1, 3.2 and from the Chebyshev ineqiality [16].

3.2.3 Information and statistical estimations for remainder and flags outputs

In the general case the distribution of probabilities of a remainder and flags outputs essentially differs from the one of probabilities of a plaintext. It's obvious, for instance, that at a uniform distribution of inputs probabilities of outputs are not distributed uniformly. In Fig. 3.3 there is a chart of the probability distribution of remainder lengths for uniformly distributed 7-byte inputs.

As before, we shall assume, that M, Y_{DT} and Y_{D} are random elements with the corresponding probability distributions. For these random elements the equalities (3.4) – (3.11) are satisfied, if we set $X = M$.

As the number of elements in the remainder range is $2^{(n-1)L+1} - 2^{rL}$, then for the entropy of the remainder $H(Y_{\mathrm{DT}})$ the following inequality is true

$$H(Y_{\mathrm{DT}}) \leq \log\left(2^{(n-1)\cdot L+1} - 2^{r\cdot L}\right). \tag{3.46}$$

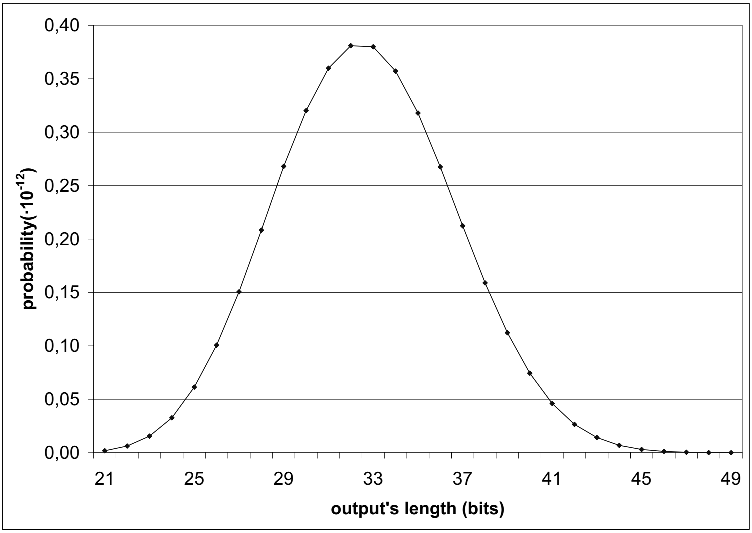

Figure 3.3: *A chart of the probability distribution of remainder lengths for uniformly distributed 7-byte inputs of a fixed MV2-transformation with the parameters* $n = 8,\ r = 3$

Similarly, the number of elements in the flags range is $(n-r+1)^L$, therefore, for the entropy of the flags $H(Y_D)$ the following inequality is satisfied

$$H(Y_D) \leq L \cdot \log(n-r+1). \tag{3.47}$$

Consider a case, when the text M is being at random and uniformly chosen from the set $\{0,1\}^{nL}$.

Let us evaluate the entropy of the remainder output in this case.

From (3.33) we have

$$H(Y_{\mathrm{DT}}) = \sum_{i=rL}^{(n-1)L} 2^{i-nL} N_i^{(c)} \log \frac{2^{nL}}{N_i^{(c)}}.$$

Due to the inequality (3.34) we have

$$H(Y_{\mathrm{DT}}) \geq 2^{-nL} \sum_{i=rL}^{(n-1)L} i \cdot 2^i N_i^{(c)} \geq (n-2+2^{r+1-n}) \cdot L.$$

Taking into account (3.46) we get the valuation:

$$\left(n-2+2^{r-n+1}\right) \cdot L \leq H(Y_{\mathrm{DT}}) < (n-1) \cdot L + 1. \quad (3.48)$$

For the conditional entropy $H(M|Y_{\mathrm{DT}})$ the equality (3.9) is true, from which we have: $H(M|Y_{\mathrm{DT}}) = H(M) - H(Y_{\mathrm{DT}})$. Then, from (3.48) it follows that

$$L - 1 \leq H(M|Y_{\mathrm{DT}}) \leq 2 - 2^{r+1-n}L. \quad (3.49)$$

Let's evaluate the entropy of flags outputs in case, when the text M is chosen at random and uniformly from the set $\{0,1\}^{nL}$. In this case $Y_{\mathrm{D}} = f(M) = f(x_1)\| \ldots \|f(x_L)$ can possess $(n-r+1)^L$ different values: $1 \leq f(x_i) \leq n-r+1$, $i = 1 \ldots L$.

Let k_j letters x_i of M have the flag output $f(x_i) = j$, where $j = 1, \ldots, n-r+1$. Then $\sum j = 1^{n-r+1} k_j = L$ and from (3.37) it follows that probability of any image at a uniform distribution of inputs will be

$$P = 2^{k_{n-r+1}} \cdot \prod_{j=1}^{n-r+1} 2^{-j \cdot k_j}.$$

Therefore ([16]), for the flag entropy we have:

$$H(Y_{\mathrm{D}}) = L! \sum_{\sum\limits_{j=1}^{n-r+1} k_j = L} \left(2^{k_{n-r+1}} \prod_{i=1}^{n-r+1} \frac{2^{-ik_i}}{k_i!}\right) \left(-k_{n-r+1} + \sum_{j=1}^{n-r+1} jk_j\right),$$

where $\quad k_i \geq 0, \; i = 1 \ldots n-r+1.$

The number of items in the sum $\sum\limits_{k_1+\ldots+k_{n-r+1}=L}$ is equal to the number of different non-negative integer solutions of the equation (3.36) and equals $\binom{n-r+L}{n-r} = \binom{n-r+L}{L}$ (see, for example, [16, chapter II]), and the concave parenthesis equals the output length. Therefore, the entropy of flags output just coincides with the mathematical expectation of the output length, and correspondingly from (3.40) it follows, that

$$H(Y_{\mathrm{D}}) = (2 - 2^{r+1-n}) \cdot L. \tag{3.50}$$

Then, from (3.5) and (3.10), in case of uniform distribution of inputs M it follows that

$$H(M|Y_{\mathrm{D}}) = H(Y_{\mathrm{DT}}|Y_{\mathrm{D}}) = (n - 2 + 2^{r+1-n}) \cdot L. \tag{3.51}$$

About distribution of bit remainder

Due to the lemma 3.1 for any remainder bit the MV2-transformation, the number of elements in a set of preimages for the value 0 is equal the one for the value 1. Therefore, the following theorem is true.

Theorem 3.1 *Let $T = (c, f)$ be an MV2-transformation, the random variable $Y_{\mathrm{DT}} \in U_{r\ n-1}$ be the remainder output, $|Y_{\mathrm{DT}}|$ be a length of the remainder output, and c_k be the k-th bit of the remainder output. Then, at a uniform distribution of inputs the probability that the value of the k-th bit is* 0 *is*

$$P\left(\mathrm{c}_k = 0 \;\middle|\; 0 \le k \le |Y_{\mathrm{DT}}|\right) = \frac{1}{2}. \tag{3.52}$$

According to the theorem 3.1 at the uniform input distribution the reminder looks like a uniform text.

3.2.4 Composition of two MV2-transformations

Let $T = (c, f) \in \mathcal{F}^r_n$ be an MV2-transformation, and let $\mathcal{M} = \bigcup\limits_{i=1}^{L} \{0,1\}^{n \cdot i}$ be a set of binary strings, and let a string $M \in \mathcal{M}$. So strings from the set $\mathcal{M}$ have lengths divisible by n. It's evident, that the remainder output $c(M)$ of the MV2-transformation of the string M, in the general case, doesn't belong to the set $\mathcal{M}$, because its length in the general case is not divisible by n. On the other hand, the remainder $c(M)$ can always be augmented from the right by a bit string $\mathbf{b}(c, M)$ so, that the concatenation $c(M)\|\mathbf{b}(c, M)$ would belong to the set of strings $\mathcal{M}$. To be certain we shall assume that $\mathbf{b}$ is either the empty string or a binary string which contains from 1 to $n-1$ zeros. Similarly one can augment a flags output. Denote by $\mathbf{A}_n$ the operation of augmentation from the right by zero bits of an arbitrary binary string till its length is divisible by n. Let two MV2-transformations: $T_1 = (c_1, f_1)$, $T_2 = (c_2, f_2) \in \mathcal{F}^r_n$. be selected at random. Let $c_i'(M) = A_n \circ c_i(M) = A_n(c_i(M))$, be a transformation being a composition of the mapping c_i and the operation $\mathbf{A}_n$, $i = 1, 2$. Then, we can define the composition of MV2-transformations in the following way:

$$T_2 \circ T_1(M) = \Big(c_2'\Big(c_1'(M)\Big),\ f_2\Big(c_1'(M)\Big)\|f_1(M)\Big). \tag{3.53}$$

Example 3.2 *Composition of two MV2-transformations. The first transformation*

$\mathbf{x}$	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
$\mathbf{c}(\mathbf{x})$	01	110	10	111	01	11	010	11	00	001	100	10	101	000	011	00
$\mathbf{f}(\mathbf{x})$	00	1	01	1	01	00	1	01	01	1	1	00	1	1	1	00

The second transformation

x	*0000*	*0001*	*0010*	*0011*	*0100*	*0101*	*0110*	*0111*	*1000*	*1001*	*1010*	*1011*	*1100*	*1101*	*1110*	*1111*
c(x)	*100*	*001*	*101*	*00*	*011*	*10*	*110*	*010*	*11*	*01*	*00*	*10*	*01*	*11*	*111*	*000*
f(x)	*1*	*1*	*1*	*00*	*1*	*01*	*1*	*1*	*00*	*01*	*01*	*00*	*00*	*01*	*1*	*1*

A plaintext

$$\begin{aligned} M = \quad & \mathit{0010}\|\mathit{0100}\|\mathit{1010}\|\mathit{1001}\|\mathit{0110}\|\mathit{1110}\|\mathit{0110}\|\mathit{1000}\| \\ & \|\mathit{1010}\|\mathit{1000}\|\mathit{1100}\|\mathit{1011}\|\mathit{0100}\|\mathit{1110}\|\mathit{1100}\|\mathit{1001} \end{aligned}$$

After the first transformation we have a remainder $C1$ and flags $F1$:

$C1 = 10\|01\|100\|001\|010\|011\|010\|00\|100\|00\|101\|10\|01\|011\|101\|001$

$F1 = 01\|01\|1\|1\|1\|1\|1\|01\|1\|01\|1\|00\|01\|1\|1\|1.$

We shall augment the remainder $C1$ by zeros from the right and get the string ($Text1$), that is the input of the second transformation.

$Text1 = 1001\|1000\|0101\|0011\|0100\|0100\|0010\|1100\|1011\|1010\|01\mathbf{00}.$

We get a new remainder and flags:

$C2 = 01\ 11\ 10\ 00\ 011\ 011\ 101\ 01\ 10\ 00\ 011,$

$F2 = 01\ 00\ 01\ 00\ 1\ 1\ 1\ 00\ 00\ 01\ 1.$

Finally we have the reminder C and flags $F = F2\|F1$:

$C =$ 0111 1000 0110 1110 1011 0000 11**00**.

$F =$ 0100 0100 1110 0000 11**00** 0101 1111 1011 0110 0011 11**00.**

Lemma 3.2 *For any string $C \in \bigcup\limits_{i=r}^{\infty}\{0,1\}^{i\cdot n}$ and for any transformation $T = (c, f) \in \mathfrak{F}_n^r$ there exists such a text $M \in \bigcup\limits_{i=r}^{\infty}\{0,1\}^{i\cdot n}$, that $c(M) = C$.*

The proof of the lemma follows directly from the possibility of presenting $i \cdot n$ in form of a sum of integer numbers $i \cdot n = \sum k_i$, where k_i such that $r \le k_i \le n - 1$.

Transformations $T \in \mathfrak{F}_n^r$ don't form a group, as the following is true

Claim 3.4 *For any three mutually different transformation*

$$T_1 = (c_1, f_1), T_2 = (c_2, f_2), T_3 = (c_3, f_3) \in \mathfrak{F}_n^r$$

there is such $M \in \bigcup_{i=r}^{\infty} \{0,1\}^{i \cdot n}$, *that*

$$c_1\Big(\mathbf{A}_n(c_2(M))\Big) \neq c_3(M).$$

The proof of the statement 3.4 follows from the lemma 3.2, as one can choose such a string M, consisting of n^2 symbols, that the length $|c_3(M)| = (n-1)n$. In this case the length

$$\Big|c_1\Big(c_2(M)\Big)\Big| < (n-1)n.$$

It follows from the statement 3.4 , that the composition of MV2-transformations is not an MV2-transformation, therefore there won't be such an MV2-transformation that would let go back to a plaintext through one round from any core obtained in the result of several transformation rounds.

Chapter 4
Main encryption scheme

4.1 A general scheme of harming based on MV2-transformations

A system of harming can be considered as a chiphersystem, a ciphertext of which consists of two or more interrelated parts.

In [37] there is an encryption method with splitting into two output channels and a device of its implementation suggested. MV2-transformations described in part 3.1 can be used as a transformer in such a device.

Each MV2-transformation $T = (c, f)$ maps an n-bit binary string x into a pair $(c(x), f(x))$, consisting of two variable length strings. It can be set by the table, in the left part of which there's a permutation of values from 1 to 2^n (see 3.1.2), and in the right part there are images consisting of "remainder" and "flag" parts.

Let us fix parameters r and n and chose an ordered set T_1, T_2, …, $T_k \in \mathcal{F}_n^r$ of random MV2-transformations. Further this set will be considered as a key.

The transformation process of an input string consist of rounds. At each round a permutation transformation and an MV2-transformation taken from the key will be performed over input data of the round. The output of an MV2-transformation is the remainder and the flags. We shall send an obtained remainder to the input of the following round, and accumulate the flags. Note, that a binary string of arbitrary length goes to the round input, and on the output we have two strings.

The number of rounds can be set directly or indirectly. As it is seen from the properties of MV2-transformations, the remainder has a length smaller than that of an input string. Therefore the threshold length for the length of the last remainder can be used as indicator stoping the transformation process. In this case rounds will be repeated till the remainder length smaller than the threshold appears.

We shall call the remainder of the last performed round as a *core.*

Such a scheme reminds a substitution permutation network (SPN) (see [15]). The architecture of SPN is a fundamental architecture of block ciphers. It is based on principles of "confusion" and "diffusion", suggested by C. Shannon [52]. These principles are implemented with the help of substitution and permutation transformations. Permutation considerably complicates interrelations between statistical and analytical characteristics of an open and an encrypted texts. Dispersion spreads influence of particular bits of an open text on as much as possible number of a ciphertext bits. It also masks statistical interrelations and complicates cryptanalysis. One of the main methods is to interleave periodically diffusion (with considerably smaller tables) and permutation in the same cipher in various combinations. Cryptographic functions are implemented by means of combinations of substitution and permutation transformation. Permutation transformations are linear, and substitution ones are the main source of non-linearity in the cipher .

A suggested method of harming can be represented in form of a global structure, represented in Fig. 4.1. The encryption process, performed according to this method, is divided into rounds. Linear and non-linear transformation alternate. Each round consists of a *linear layer* and a *none-linear layer.*

Using a round procedure $Raund(M)$, can be probabilistic and be performed according to the scheme

$$Raund(M) = \Big(R||C(K,R,M)\,,\,F(K,R,M)\Big),$$

where R is a randomly generated bit block, $C(K,R,M)$ and $F(K,R,M)$ are the first and the second output components of the substitution transformation, M is a round input and K is a key.

The remainder output of a round is $R||C(K,R,M)$.

During decryption a deterministic procedure $Raund^{-1}$ works according to the following recurrent scheme:

$$(R_i||C_i, F_i) = Raund^{-1}\Big(R_{i+1}||C_{i+1}\,,\,F_{i+1}\Big) \text{ for } 0 < i < P$$

$$\text{and } (M,\Lambda) = Raund^{-1}\Big(R_1||C_1\,,\,F_1\Big),$$

where Λ – is the empty string.

In the architecture of the offered scheme there's a significant difference from the architecture of SPN of block ciphers. At performing each round the whole text, rather than one block, is processed.

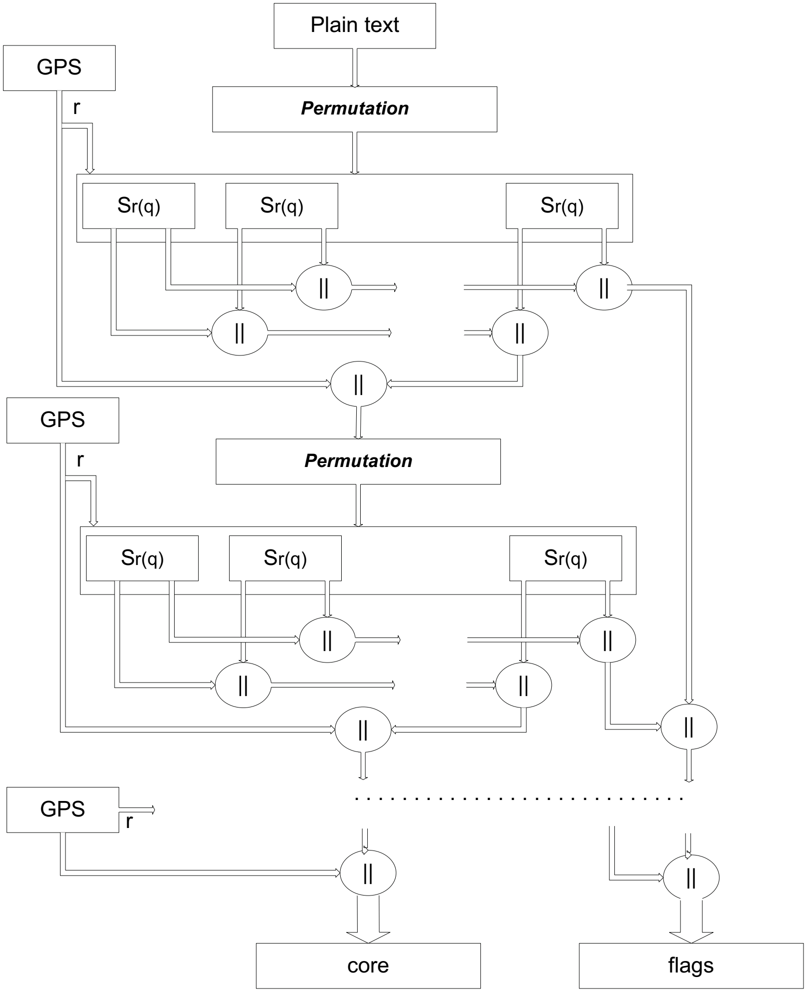

Figure 4.1: *Presentation of the global structure as SPN. Here GPS — Random number generator, Permutation – a linear transformation, Sr(q) – a substitution transformation.*

4.2 Preliminary analysis of the general scheme which uses MV2-transformations

It is accepted that evaluation of cipher security is executed by building attacks or by indirect features. At that in traditional ciphers at least a ciphertext is considered to be known, and the task of a cryptanalyst is to recover the key or the plaintext.

For schemes of harming, similar to those considered above, a ciphertext consists of two parts – a core and flags. Therefore, to use such schemes it's necessary to consider additional variants when only a core or a core and keys are known, or only flags or flags and keys are known.

4.2.1 Round of the general scheme of harming

As we have already mentioned, the whole process of harming in the suggested scheme is divided into rounds. Each round consists of permutation and substitution transformations. A permutation transformation provides bit dispersion of a plaintext which goes to the input of a round. An ideal variant is a bit permutation of the entire plaintext. But the authors don't know the algorithms that would implement such a permutation transformation. Therefore, we use a local permutation. One of the variants is the following one: a input string is divided into l-bit blocks, and a transformation is carried out under each of them.

After a permutation transformation one of MV2-transformations is carried out, which is chose from the key according to a random value R, that is obtained from a RNG. The outputs of this transformation are the remainder and the flags. As the remainder goes to the input of the next round, its length should be divisible by n bit and a binary string should be added to it in the general case.

Evaluation of output lengths

Let a message M, consisting of L n-bit binary strings (codes of symbols of a plaintext) go to the input of the device. We denote mathematical expectations of the number of symbols (n-bit binary strings) by $\mathbf{E}(L_m^{(c)})$ and $\mathbf{E}(L_m^{(f)})$, accordingly, the cores and the total flags after executing m rounds of the transformation.

If a random binary string consisting of L_{i-1} n-bit substrings goes to the input of the ith round, then from the statement 3.3, for the number of n-bit strings in the remainder output $L_i^{(c)}$ and that of the flag output $L_i^{(f)}$ we have:

$$L_i^{(c)} \approx \mathrm{K_c} \cdot L_{i-1}^{(c)},$$
$$L_i(f) \approx \mathrm{K_f} \cdot L_{i-1}^{(c)},$$

where (3.41) and (3.42) are coefficients introduced in the definitions 3.2 and 3.3.

Let's assume that s n-bit binary strings are added to every remainder, as strings obtained after an MV2 transformation should be augmented by bits till their length is divisible by n, and as in the general scheme it can be required to transmit the number of the transformation chosen from the key. Therefore,

$$L_i^{(c)} \approx \mathrm{K_c} \cdot L_{i-1}^{(c)} + s.$$

Similarly for the flags of the i-th round we have

$$L_i^{(f)} \approx \mathrm{K_f} \cdot L_{i-1}^{(c)} + 1.$$

Then at the m-th round (for great enough m) an estimated number of symbols in the remainder is:

$$L_m^{(c)} = \mathrm{K_c}^m \cdot \left(L + s \cdot \sum_{i=1}^{m} \mathrm{K_c}^{-i} \right) = \mathrm{K_c}^m \cdot L + s \cdot \frac{1 - \mathrm{K_c}^m}{1 - \mathrm{K_c}}.$$

From which, using (3.43), we have

$$L_m^{(c)} \approx \mathrm{K_c}^m \cdot L + \frac{1}{\mathrm{K_f}} \cdot s. \tag{4.1}$$

Then, an estimated number of symbols in the flag output will be:

$$L_m^{(f)} = \mathrm{K_f} \cdot L + \mathrm{K_f} \cdot \sum_{i=1}^{m-1} \left(L \cdot \mathrm{K_c}^i + \frac{1 - \mathrm{K_c}^i}{\mathrm{K_f}} \cdot s \right) =$$

$$= (1 - \mathrm{K_c}^m) L + \left(m - 1 - \frac{1 - \mathrm{K_c}^m}{\mathrm{K_f}} \right) \cdot s.$$

From where

$$L_m^{(f)} \approx L + m \cdot s. \tag{4.2}$$

If at encryption there is a requirement, that a number of n-bit symbols in the core output shouldn't exceed a threshold L_c, then, due to (4.1), an estimated number of rounds m_L is roughly:

$$m_L \approx 1 + \frac{\log L_c - \log L}{\log \mathrm{K_c}}. \tag{4.3}$$

From the statement 3.3 it follows, that at executing a round of a transformation under a text the length of which is L bytes with the probability no less than $1 - L^{-2}$ the number of n-bit symbols in the obtained remainder will be within the limits of :

$$(\mathrm{K_c} - \sigma_c/n) \cdot L \;\leq\; |C|/n \;\leq\; (\mathrm{K_c} + \sigma_c/n) \cdot L,$$

and the number of n-bit symbols in the obtained flags:

$$(\mathrm{K}_\mathrm{f} - \sigma_f/n) \cdot L \;\le\; |F|/n \;\le\; (\mathrm{K}_\mathrm{f} + \sigma_f/n) \cdot L,$$

where σ_c and σ_f are root-mean-square deviations of lengths of output texts from evaluated average values.

Evaluation of a number of texts when a core is known and flags are unknown

If a round permutation is fixed, then a set of texts giving the same remainder is only dependent on the substitution transformation $T \in \mathcal{F}_n^r$. As it is shown in 3.2, for a known remainder C_i of the length $|C_i| = l$ bit, the number of its preimages for one round is determined by the expression (3.33).

Thus, if only the core is known, and there's no limitations for the number of rounds, then even if the keys are known there's an infinite set of texts giving such the core. At a limited number of rounds a set of plaintexts that corresponds to the given core is finite.

If an random text went to the input of every round, then according to the theorem 3.1, it would be possible to consider the remainder output as a uniform sequence. Let C be a core after executing m rounds and let N_m be the number of preimages of C. Then, to evaluate the number of preimages of N_m one can use the inequality (3.49). Due to this inequality we have:

$$\log N_m > \frac{1}{\mathrm{K}_\mathrm{c}}\frac{|C|}{n} + \ldots + \frac{1}{{\mathrm{K}_\mathrm{c}}^m}\frac{|C|}{n},$$

and

$$N_m \ge 2^{\frac{1-{\mathrm{K}_\mathrm{c}}^m}{\mathrm{K}_\mathrm{f}}\cdot\frac{|C|}{n}}. \tag{4.4}$$

Let N_L be a number of texts corresponding to the known core. Accordingly, if L is a number of symbols in a plaintext received at the system's input is known, then for the evaluation of N_L from the inequality (4.4) we have

$$N_L \geq 2^{\frac{1-\mathrm{K_c}^m}{\mathrm{K_f}} \cdot L}. \tag{4.5}$$

4.2.2 Key

The key is an ordered set T_1, T_2, $\ldots$, $T_k \in \mathfrak{F}_n^r$ of MV2-transformations. Every transformation can be set by a permutation numbers from 0 to $2^n - 1$ (see 3.1.2). Consequently, the number of keys in key space is $(2^n!)^k$. Note, that the key T_1, T_2, $\ldots$, T_k can be generated from a short binary string (a master of key). In this case the length of a master key can be up to $k \cdot \log(2^n!)$ bits.

A round substitution transformation can be chosen from the key by both a determined rule and randomly. If a substitution transformation is chosen at random at every round, the general scheme will be a probabilistic cipher. But in this case it's necessary to transmit a reference to a transformation chosen from the key; it can be done, for instance, in the remainder output.

It's significant to note, that in the mentioned scheme a substitution transformation is defined at a set of 2^n elements, and a number of various symbols from the alphabet $\{0, 1\}^n$ being used in an input text may be considerably less than 2^n. It means that at performing $m < k$ rounds the key won't be entirely used, and, consequently, is too big. If a substitution transformation is chosen at random at every round, the key will be entirely used at long life time.

4.2.3 Randomization in the general scheme of harming

Determinate cryptosystems have leak of information; for example it's easy for an adversary to determine a situation when the same message is sent repeatedly. Another disadvantage of these systems becomes apparent when little message space is used. A brute force attack is possible in this case. If a plaintext has a lot of ciphertexts it leads to additional indeterminacy at a cryptanalysis.

In the suggested scheme accumulation of all output flags obtained at each round takes place. It's obvious, that a total size of the flags of all rounds may be close to the size of the plaintext at performing great enough number of rounds. Though the statistics of flags is considerably different from the one of an initial text, nevertheless a question emerges: whether it's possible to restore information according to known flags only, for instance, on the basis of the analysis of frequency of symbols, especially at known keys. For some families of texts this question can be answered positively, that gives concern for cryptographic resistance of the suggested scheme.

In ciphersystems randomization of a plaintext [29] is used to counteract attacks based on frequency analysis. Randomization is an old method and can be performed in different ways. One of them is whitening from a random (pseudorandom) number generator (RNG). A stream cipher, for example, can be used as such a generator.

Using a stream cipher for whitening, firstly, leads to substitution of a message alphabet for an alphabet of this cipher, and, secondly, counteracts attacks based on frequency analysis.

A general scheme also allows performing randomization of a cipher if a substitution transformation is randomly chosen from the key at every round. For this purpose a random (pseudorandom) number generator (RNG) can be used. The nature of a generator doesn't matter for a cipher, therefore any RNG, including a physical one, can be used.

When using a good GPS for the same plaintext k^m various output texts are possible if m encryption rounds are performed.

4.3 Two channel encryption algorithm MV2

We have already mentioned that popular modern computer systems work with data the minimal addressable unit of which is a byte that equals eight bit. Consequently, it's reasonable to choose a parameter n, divisible by 8: 8, 16, 32, ... and so on at building a device of harming designed to work in such systems.

On the basis of the scheme of harming suggested in 4.1 we created a symmetric probabilistic cipher which implements the universal mechanism of harming – the MV2 algorithm.

In this cipher encryption is executed according to the following algorithm:

Encryption

Input: a plaintext $M\,(8 \times L(bits))$
a secret key $K = T_1, \ldots, T_k, T_i \in \mathfrak{F}_8^3$
a number of rounds m

```
BEGIN
    $C_0 := M$; $F := \Lambda$; generate a pseudorandom number $R_0$;
    using $R_0$ generate an index $j$ of substitution transformation;
    $C_0 := R_i \| (C_0 \oplus GPS(T_j))$; // whitening of the plaintext
    for $i = 1$; to $i = m$ do // $P$ rounds are executed;
        1. $C_{i-1} :=$ Mix$(C_{i-1})$ // a remainder of a previous round is permuted block by block.
        2. Generate a pseudorandom number $R_i$.
        3. Generate (using $R_i$) an index $j$ of a transformation.
        4. $\Big(C_i, F_i\Big) := T_j(C_{i-1})$ // the substitution transformation $T_j$ is carried out. In the result we obtain a new remainder $C_i$ and flags $F_i$.
        5. $C_i = R_i \| C_i$; $F = F_i \| F$.
    end(for).
    $C = C_m$;
END.
```

Output: a ciphertext $\Big(C\,,\,F\Big)$.

Decryption is executed according to the following algorithm:

Decryption

Input: a ciphertext $\Big(C\,,\,F\Big)$

a secret key $K = T_1, \ldots, T_k,\ T_i \in \mathfrak{F}_8^3$

```
BEGIN
    $C_m := C$;
    While $(F \neq \Lambda)$ do
        1. Extract $R$ from $C$; computer the index $j$ from $R$;
        2. Moving along $C_m$ and $F$, choose pairs of images
           $(c_i, f_i), i = 0, 1, ...$
        3. For each pair $(c_i, f_i),\ i = 0, 1, ...$ find a preimage
           $x_i$; set $C := x_0 \| x_1 \| ...$ .
        4. We assign $F$ a new position.
        5. If $(F = \varnothing) AND$ position $C_m \neq EOF$ then $C =$
           'error message'.
        6. $C :=$ ReMix$(C_m)$ // reshuffle the result.
    End(while).
    If ( no 'error message') Begin
        Extract $R$ from $C$; computer the index $j$ from $R$;
        $C := C_m \oplus GPS(T_j)$ // whitening of the plaintext
        is removed;
    end (Begin)
    $M = C$;
END;
   Output: Error message
           or a plaintext    $M\,(8 \times L)$ bits.
```

This algorithm describes a general method of harming for byte-oriented texts. The exact implementation of the MV2 algorithm and results of testing are described in 7.

Chapter 5 Statistical properties of substitutional transformations

5.1 S-box Properties

Since the S-boxes comprise the only nonlinear component of an SPN, they are a crucial source of cryptographic strength. S-box research has focused largely on determining which properties yield a cryptographically "good" S-box. Some of the important properties are given below. In this section, we use $\mathbf{e}_i$ to denote a unit vector with 1 in position i, and $w(\mathbf{v})$ to mean the Hamming weight of vector $\mathbf{v}$.

5.1.1 Completeness

In 1979, Kam and Davida defined the property of completeness for a bijective function $f : \{0,1\}^t \rightarrow \{0,1\}^t$: f is complete if for all $i, j \in \{0, 1, \ldots, t-1\}$, there exists $x \in \{0,1\}^t$ such, that $f(x)$ and $f(x \oplus e_i)$ differs in at least bit j (e_i is the t-bit unit vector with a 1 in position i). That is to say, every output bit depends upon every input bit. An S-box is complete if it satisfies this property, and an SPN is complete if it is a complete function from $\{0,1\}^N$ to $\{0,1\}^N$ for every key. Kam and Davida gave an algorithm for constructing complete S-boxes, and specified a permutation to be used in each round in order to achieve completeness after a minimum number of rounds.

In research published in 1981 and 1982, F. Ayoub extended Kam and Davida's study of permutations which produce complete SPNs. He introduced a class of permutations, referred to as cryptographically equivalent permutations (CEP). If a permutation belongs to CEP, and if $N = an^2$ for some $a \in \{1, 2, \ldots, n\}$, then an N-bit SPN constructed using this permutation and complete $n \times n$ S-boxes will satisfy completeness after a minimum number of rounds (two if $a = 1$ and three if $2 \leq a \leq n$).

Ayoub also investigated the possibility of using the key to construct a random permutation used in all the rounds, and proved that there is a high probability that the resulting SPN will be complete after a minimum number of rounds. Moreover, Ayoub suggested the possibility of designing SPNs in which both the S-boxes and the permutation are generated in a pseudo-random fashion from the key, and claimed that such an approach would not weaken the security of the SPN. We investigate the idea of using key-dependent S-boxes in an SPN.

5.1.2 Bit Independence

Webster and Tavares, in the paper in which they introduced SAC [53], also defined a property called the *bit inde-*

pendence criterion (BIC). A function $f : \{0,1\}^t \to \{0,1\}^t$ satisfies BIC if for all $i, j, k \in \{0, 1, \ldots, t-1\}$ with $j \neq k$, inverting input bit i causes output bits j and k to change independently. We say f satisfies *maximum order* BIC (MO-BIC) if the same output bit independence holds whenever an input change consisting of one or more bits occurs.

5.1.3 Nonlinearity

A function $f : \{0,1\}^t \to \{0,1\}$ is called *affine* if there exist constants $a_i \in \{0,1\}$, for $i = 0, 1, \ldots, t$, such that for all $X = x_{t-1} \ldots x_1 x_0 \in \{0,1\}^t$,

$$f(X) = a_t \oplus a_{t-1}x_{t-1} \oplus \cdots \oplus a_1 x_1 \oplus a_0 x_0.$$

An affine function is called *linear* if $a_t = 0$. S-boxes with "high nonlinearity" are needed to make an SPN immune to *linear cryptanalysis*, introduced in 1993 by M. Matsui [32]. Linear cryptanalysis attempts to find a linear equation relating plaintext, ciphertext and key bits, i.e., it looks for indices $i_1, i_2, \ldots, i_b, j_1, j_2, \ldots j_c$, and $l_1, l_2, \ldots, l_d$, such that

$$P_{i_1} \oplus P_{i_2} \oplus \cdots P_{i_b} \oplus C_{j_1} \oplus C_{j_2} \oplus \cdots \oplus C_{j_c} = K_{l_1} \oplus K_{l_2} \oplus \cdots \oplus K_{l_d}.$$

$$P_{i_1} \oplus P_{i_2} \oplus \cdots P_{i_b} \oplus C_{j_1} \oplus C_{j_2} \oplus \cdots \oplus C_{j_c} = K_{l_1} \oplus K_{l_2} \oplus \cdots \oplus K_{l_d}.$$

If the bits in this equation are assigned at random, the equation will be satisfied with probability exactly 1/2. Matsui's attack exploits the situation in which this equation is satisfied with probability significantly more or less than 1/2.

We can quantify the term ”high nonlinearity” as follows. Let A_t be the set of all affine functions $g : \{0,1\}^t \to \{0,1\}$. For $f : \{0,1\}^t \to \{0,1\}$, we define the *nonlinearity* of f, $nl(f)$ as

$$nl(f) = \min_{g \in A_t} wt(f \oplus g)$$

(in this expression, we view f and g as 2^t-bit vectors). Clearly $nl(f)$ measures the distance of f from the closest affine function. If S is an S-box, let L be the set of all linear combinations of the columns of S. Then the nonlinearity of S is

$$nl(S) = \min_{l \in L \setminus 0} nl(l).$$

It is not hard to see that all vectors $l \in L \setminus 0$ and $g \in A_t \setminus 0$ are balanced. It follows that $nl(l)$ is always even. Also, if $g \in A_t \setminus 0$ is a linear function, and $\overline{g}$ is the affine function which is the bitwise complement of g, then

$$wt(l \oplus \overline{g}) = 2^n - wt(l \oplus g),$$

and therefore $nl(S) \in \{0, 2, 4, \ldots, 2^{n-1}\}$.

In 1983, Gordon and Retkin [18] showed that the probability that a randomly chosen invertible $n \times n$ S-box has one or more output bits which are linear functions of the input bits is

$$\frac{2n(2^n - 1)(2^{n-1}!)^2}{2^n!}.$$

Experimental results support the theoretical result. For example, Heys [20] generated 200 random invertible 8×8 S-boxes and found that each satisfied $86 \leq nl(S) \leq 98$.

5.1.4 XOR Table Distribution

In 1991, Biham and Shamir introduced a powerful cryptanalytic technique known as differential cryptanalysis [4]. They have successfully applied their attack to a variety of SPNs. Differential cryptanalysis requires knowledge of the XOR tables of the S-boxes. For an $n \times n$ S-box, S, the XOR table of

S has rows and columns indexed by $0, 1, \ldots, 2^n - 1$, and the table entries are defined as follows. If $i, j \in \{0, 1, 2, \ldots, 2^n - 1\}$, position $[i, j]$ in the XOR table p contains the value

$$p(i, j) = |\{X \in \{0, 1\}^n : S(X) \oplus S(X \oplus i) = j\}|$$

(we are treating i and j as their equivalent n-bit strings). It can be shown that $p(i, j)$ always evaluates to an even number. The pair (i, j) is called an input/output XOR pair. Differential cryptanalysis exploits such XOR pairs with large XOR table entries. An SPN can be secured against differential cryptanalysis by selecting S-boxes with low XOR table entries, ideally all 0 or 2 (the one exception is entry (0,0) which has value 2^n). Even if the XOR table is not directly calculated, resistance to differential cryptanalysis can be achieved by assuring that the S-boxes have good *diffusive* properties, i.e., they reasonably satisfy AVAL or SAC.

If S is a randomly chosen invertible $n \times n$ S-box, and $0 \leq A \leq 2^n$ is an even integer, a formula of Youssef and Tavares [56] gives an upper bound on the probability that the maximum XOR table entry of S (denoted maxXOR(S)) is $\geq A$. For example, if $n = 8$ and $A = 16$, we have $prob[maxXOR(S) \geq 16] \leq 0.0042$.

5.2 Properties of Random S-boxes

Since the object of this proposal is a new cipher using key-dependent S-boxes, it will be useful to investigate the average properties of random invertible $n \times n$ S-boxes. A series of combinatorial results have demonstrated that a randomly chosen s-box of sufficient size will possess several of these desirable properties with high probability.

5.2.1 Completeness

O'Connor proved that a randomly chosen $n \times n$ invertible S-box has a high probability of being complete for sufficiently large n. In fact, he showed that the probability that such an S-box is not complete is

$$o\left(\frac{\sqrt{2^n}}{2^{2^{n-1}+n+1}}\right).$$

For an exact formula, see [34].

5.2.2 Avalanche and Strict Avalanche

The authors have not found any results giving the probability that a random invertible $n \times n$ S-box satisfies AVAL or SAC (although there are bounds on the probability that a random function $f : \{0,1\}^t \rightarrow \{0,1\}$ satisfies SAC). On the other hand, a number of theoretical and experimental results exist concerning the AVAL property for SPNs. Heys and Tavares developed a probabilistic model for the AVAL property of an SPN. Their results for $N = 64$ and $n = M = 8$, using randomly selected S-boxes and the fixed permutation of Kam and Davida, indicate that AVAL is reasonably satisfied after 5 or more rounds. In fact, if E_R is the expected number of output bit changes after R rounds when one input bit is flipped, and we define $\varepsilon = |1 - E_R/(N/2)|$, then $\varepsilon \leq 10^{-5}$ for $R \geq 7$.

5.2.3 Nonlinearity

For an invertible $n \times n$ S-box S and an integer $2L(0 \leq L \leq 2^{n-2})$, Youssef and Tavares prove that

$$\mathbf{prob}[nl(S) \leq (2^{n-1} - 2L)] < \\ < \frac{2(2^{n-1}!)^2(2^n-1)^2}{2^n!} \sum_{i=L}^{2^{n-2}} \binom{2^{n-1}}{2^{n-2}+i}^2 .$$

If $n = 8$, we have, for example, $\mathbf{prob}[nl(S) \leq 64] < 1.4 \cdot 10^{-11}$ and $\mathbf{prob}[nl(S) \leq 80] < 4.6 \cdot 10^{-5}$. Experimental results support the theoretical result of (3). For example, Heys generated 200 random invertible $n \times n$ S-boxes and found that each satisfied $86 \leq nl(S) \leq 98$.

5.2.4 XOR Table Distribution

If S is a randomly chosen invertible $n \times n$ S-box, and $0 \leq A \leq 2^n$ is an even integer, a formula of Youssef and Tavares gives the probability that the maximum XOR table entry of S (denoted maxXOR(S)) is $\geq A$. For example, if $n = 8$ and $A = 16$, we have

$$\mathbf{prob}[\mathrm{maxXOR}(S) \geq 16] \leq 0.0042.$$

5.2.5 Cyclic Properties

There is some indication that the cyclic properties (cycle length, number of cycles) of an S-box are related to other cryptographic properties. Youssef et al. give experimental results which show that, on average, S-boxes with fewer fixed points have higher nonlinearity and lower maximum XOR table entries. They prove that the expected number of fixed points for a random invertible S-box is 1, with a variance of 1. They also state that the expected value and variance of the number of cycles is approximately $\ln 2^n \approx 0.69n$; and that the expected cycle length is $2^{n-1} + 1/2$, where the expected cycle length is defined as the value of the length of the cycle to which a randomly chosen element belongs.

5.3 Dependence criteria

For substitution-permutation networks usually use some criteria what is known as dependence criteria. They are destined for evaluation a security S-Boxes. In this section, we shall state the definitions as given in [48].

For a binary string $x = (x_1, \ldots, x_n) \in (GF(2))^n$, the binary string $x^{(i)} \in (GF(2))^n$ denotes the binary string obtained by complementing the i-th bit of x (for $i = 1, \ldots, n$). The Hamming weight $w(x)$ of x is defined as the number of nonzero components of x. A function $f : (GF(2))^n \to (GF(2))^m$ of n input bits into m output bits is said to be complete, if each output bit depends on each input bit, i.e.

$$\forall i = 1, \ldots, n \, \forall j = 1, \ldots, m \, \exists x \in (GF(2))^n$$

with $\left(f(x^{(i)})\right)_j \neq \left(f(x)\right)_j$

Feistel [15] has defined a property of a s-block and SPN called *avalanche effect* (AVAL). A function $f : (GF(2))^n \to GF(2))^m$ satisfies AVAL property if whenever one input bit is changed, an average half of output bits change, i.e.,

$$\frac{1}{2^n} \sum_{x \in (GF(2))^n} w(f(x^{(i)}) - f(x)) = \frac{m}{2} \quad \text{for all} \quad i = 1, \ldots, n.$$

In 1985, Webster and Tavares combined the completeness and avalanche properties into the *strict avalanche criterion* (SAC). A function $f : \{0,1\}^t \to \{0,1\}^t$ satisfies SAC if for all $i, j \in \{0, 1, \ldots t-1\}$, flipping input bit i changes output bit j with probability exactly 1/2. It is easy to demonstrate that a function f which satisfies SAC is complete, and satisfies AVAL.

A function $f : (GF(2))^n \to GF(2))^m$ satisfies SAC if after changing one input bit, each output bit changes with probability 1/2, i.e.

$$\forall\, i = 1, \ldots, n \ \ \forall\, j = 1, \ldots, m \ \ Pr((f(x^{(i)}))_j \neq (f(x))_j) = \frac{1}{2}.$$

The *dependence matrix* of the function $f : (GF(2))^n \to GF(2))^m$ is an $n \times m$ matrix A with elements a_{ij} equal to the number of inputs for which complementing the i-th input bit results in a change of the j-th output bit, i.e.

$$a_{ij} = \#\{x \in (GF(2))^n \mid (f(x^{(i)}))_j \neq (f(x))_j\} \tag{5.1}$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, m$.

The *distance matrix* of a function $f : (GF(2))^n \to GF(2))^m$ is $n \times (m+1)$ matrix B with elements b_{ij} equal to the number of inputs, for which complementing the i-th input bit results in a change of j output bits, i.e.

$$b_{ij} = \#\{x \in (GF(2))^n \mid w(f(x^{(i)}), f(x)) = j\} \tag{5.2}$$

for $i = 1, \ldots, n$ and $j = 0, \ldots, m$.

Obviously, at the input size $n > 30$ due to limits of memory resources it's not possible to compute matrices of dependence and distance for all possible inputs. Therefore, one usually considers a "suitable" number of randomly chosen inputs. The dependence and distance matrices are then defined as follows:

$$a_{ij} = \#\{x \in \mathcal{X} \mid (f(x^{(i)}))_j \neq (f(x))_j\} \tag{5.3}$$

for $i = 1, \ldots, n$, $j = 1, \ldots, m$, and

$$b_{ij} = \#\{x \in \mathcal{X} \mid w(f(x^{(i)}), f(x)) = j\}, \tag{5.4}$$

where $\mathcal{X}$ is a "suitable" randomly chosen subset of $(GF(2))^n$.

Assume we've computed the dependence matrix A and the distance matrix B of a function $f : (GF(2))^n \to GF(2))^m$ for a set of inputs X, where X is either $(GF(2))^n$, or a random subset of $(GF(2))^n$.

The *degree of completeness* of a function f is defined as

$$d_c = 1 - \frac{\#\{(i,j)|a_{ij} = 0\}}{nm}. \tag{5.5}$$

The *degree of avalanche effect* of a function f is

$$d_a = 1 - \frac{1}{nm} \cdot \sum_{i=1}^{n} \left| \frac{2}{\#\mathcal{X}} \sum_{j=1}^{m} j \cdot b_{ij} - m \right|. \tag{5.6}$$

The *degree of strict avalanche criterion* of a function f is defined as

$$d_{sa} = 1 - \frac{1}{nm} \cdot \sum_{i=1}^{n} \sum_{j=1}^{m} \left| \frac{2a_{ij}}{\#\mathcal{X}} - 1 \right|. \tag{5.7}$$

For the function f to have good degrees of completeness, avalanche effect, and strict avalanche criterion, the numbers d_c, d_a, and d_sa must satisfy

$$d_c = 1,\ d_a \approx 1,\ d_{sa} \approx 1.$$

It is known [24], that affine and linear functions don't satisfy a strict avalanche criterion. Nowadays it's considered that a good cryptographic function should satisfy a strict avalanche criterion ([24, 25, 48] etc.).

5.4 Dependence criteria for mappings with variable length outputs

5.4.1 Metric at a set of functions with images of various length

As is well known, a metric at a set $\mathcal{X}$ is a function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, possessing the following features:

1. for any $x, y \in \mathcal{X}$ $d(x, x) = 0$ if and only if $x = y$;
2. for any $x, y \in \mathcal{X}$ the following is performed: $d(x, y) = d(y, x)$ (symmetry);
3. for any $x, y, z \in \mathcal{X}$ the triangle inequality is performed:

$$d(x, y) \leq d(x, z) + d(z, y).$$

Usually a metric known as the Hamming distance is used at research of mappings with a fixed image length at a set $\{0, 1\}^n$:

Let x and y be n-bit binary strings, where $\mathrm{x}_i, \mathrm{y}_i \in \{0, 1\}$ define the corresponding i-th binary bit.

Definition 5.1 *The Hamming distance between the binary strings x and $y \in \{0, 1\}^n$ is the number:*

$$w(x, y) = \sum_{i=1}^{n} \mathrm{x}_i \oplus \mathrm{y}_i. \tag{5.8}$$

At a set of binary strings of variable length one can introduce a metric similar to the Hamming distance.

Definition 5.2 *The distance between the binary strings x and $y \in \mathfrak{U}_{rm}$ is the number*

$$h(x, y) = w(\overline{x}^k, \overline{y}^k) + \big||x| - |y|\big|, \tag{5.9}$$

where $k = \min\{|x|, |y|\}$, and $\overline{x}^k$ and $\overline{y}^k$ denote k-bit binary strings, the corresponding bits of which coincide with the corresponding bits of the binary strings x and y.

Proof of correctness of introducing a metric. We shall prove that the function h introduced in the definition 5.2 is a metric at the set $\mathfrak{U}_{rm}$.

It's obvious, that the first and the second points of the metric definition are performed for the function h, therefore to prove h is a metric at a set of binary strings of various length, it's necessary to prove performing of triangle inequality.

We shall take arbitrary binary strings x, y, z. Let $k = \min\{|x|, |y|, |z|\}$.

Denote through $\overline{x}^k$, $\overline{y}^k$ and $\overline{z}^k$ – k-bit binary strings, the corresponding bits of which coincide with the corresponding bits of the binary strings x, and y and z.

The Hamming distance w is a metric, therefore the following is performed:

$$w(\overline{x}^k, \overline{y}^k) \leq w(\overline{x}^k, \overline{z}^k) + w(\overline{z}^k, \overline{y}^k).$$

Lengths of the binary strings – $|x|$, $|y|$ and $|z|$ are integer nonnegative numbers, therefore the triangle inequality is true for them :

$$\big||x| - |y|\big| \leq \big||x| - |z|\big| + \big||z| - |y|\big|.$$

As

$$h(x,z) = w(\overline{x}^k, \overline{z}^k) + \big||x| - |z|\big|,$$
$$h(y,z) = w(\overline{z}^k, \overline{y}^k) + \big||z| - |y|\big|.$$

then

$$h(x,y) \leq h(x,z) + h(z,y).$$

$\square$.

5.4.2 Defining dependence criteria for substitution transformations with a variable length output

The expressions (5.5), (5.6) and (5.7) are cited for the mappings $f : \{0,1\}^n \rightarrow \{0,1\}^m$. An MV2-transformation is used as a substitution transformation in the general scheme. This transformation is a pair of mappings with images of various length $g : \{0,1\}^n \rightarrow \bigcup\limits_i \{0,1\}^i$. In general, output lengths don't coincide for such a transformation, i.e.:

$$g(x_1) = y_1 \in \{0,1\}^i \,,\, g(x_2) = y_2 \in \{0,1\}^j,\ i \neq j.$$

The Hamming distance is used in dependence criteria to measure the degree of difference of binary strings. Using dependence criteria for research of the general scheme of harming requires reconsidering definitions and formulae for computing distance and dependency matrixes. We defined its analogue for variable length mappings (see the definition 5.2 from i. 5.4.1).

During the test we shall use the following formulae to correspond to dependence criteria:

$$b_{ij} = |\{x \in X \mid h(g(x^{(i)}), g(x)) = j\}|, \tag{5.10}$$

$$\overline{d}_a = 1 - \frac{1}{mn} \cdot \sum_{i=1}^{n} \left| \frac{2}{\#\mathfrak{X}} \sum_{j=1}^{\overline{m}_i} j \cdot b_{ij} - \overline{m}_i \right|, \tag{5.11}$$

$$\overline{d}_{sa} = 1 - \frac{1}{mn} \cdot \sum_{i=1}^{n} \sum_{j=1}^{\overline{m}_i} \left| \frac{2a_{ij}}{\#\mathfrak{X}} - 1 \right|, \tag{5.12}$$

where $\overline{m}_i = \max\{|g(x^{(i)})| : x \in X\}$ – the maximal output length at changing the i-th bit, and $m = \max\limits_i\{\overline{m}_i\}$.

Similarly, in the expressions (5.11) and (5.12) instead of the maximal lengths of $\overline{m}_i$ output one can take the average output lengths:

$$\widetilde{m}_i = \frac{1}{\#\mathfrak{X}} \sum_{x \in \mathfrak{X}} |g(x^{(i)})|.$$

In this case the expressions (5.11) and (5.12) will look like:

$$\widetilde{d}_a = 1 - \frac{1}{mn} \cdot \sum_{i=1}^{n} \left| \frac{2}{\#\mathfrak{X}} \sum_{j=1}^{\widetilde{m}_i} j \cdot b_{ij} - \widetilde{m}_i \right|, \tag{5.13}$$

$$\widetilde{d}_{sa} = 1 - \frac{1}{mn} \cdot \sum_{i=1}^{n} \sum_{j=1}^{\widetilde{m}_i} \left| \frac{2a_{ij}}{\#\mathfrak{X}} - 1 \right|, \tag{5.14}$$

where $m = \max\limits_i\{\widetilde{m}_i\}$.

In Fig. 5.1 there are charts of results of testing the basic implementation of the MV2 encryption algorithm (see Application 7) for correspondence to a strict avalanche criteria. One can see from the chart that values computed at using the average lengths according to the formulae (5.14) are greater than those computed according to the formulae (5.12) at using the maximal output lengths.

The expressions (5.11, 5.12) and (5.13, 5.14) reflect more precisely the specific character of the transformation, than

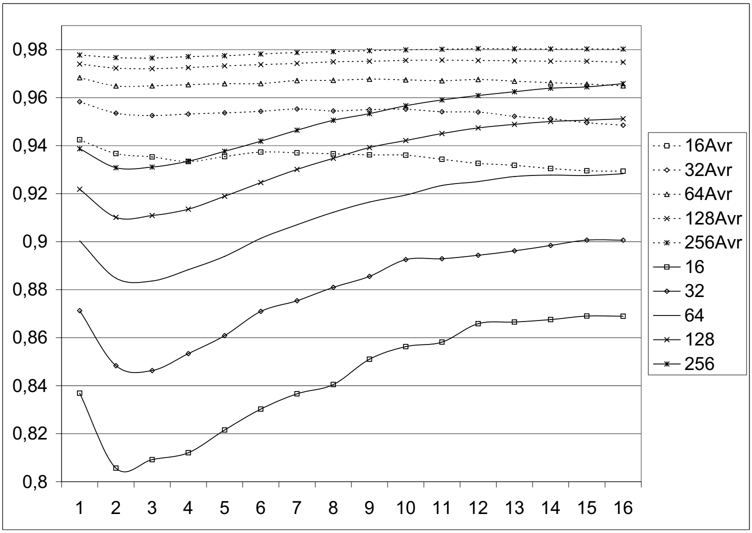

Figure 5.1: *Dependence of the degree of a strict avalanche criterion* $\overline{d}_{sa}$ *and* $\widetilde{d}_{sa}$ *(Avr) on a number of rounds at different output lengths (16, 32, 64, 128 and 256 bytes)*

(5.6) and (5.7), but still add some error in the "tails" of strings of distance and dependence matrices.

It's connected to the fact, that in the basic expressions (5.6) and (5.7) the normalization coefficient $1/mn$ due to the mapping properties is used correctly. Therefore, the absolute values of the degrees d_a and d_{sa} reflect quality of cryptographic transformations. In case with variable length transformations the normalization coefficients used in formulae (5.11), (5.12), (5.13) and (5.14) are not exactly correct. For different classes of transformations at such normalizations various hard errors are included, which make the degree value less. In the given case these criteria can serve for comparison of cryptographic transformations of the same class.

In 7 the results of testing of one of the implementations of the general scheme for compliance with dependence criteria are given. The analysis of the results (see 7.4) let choose permutation transformations, evaluate influence of whitening, and the value of a pseudorandom choice of substitution transformations. Besides, testing for compliance with dependence criteria confirmed the supposition about increase of the algorithm efficiency at increasing a plaintext length.

Chapter 6
Security Models

To precisely quantify the *security* of a cipher, and thus prove that it fulfills given security requirements, is an extremely difficult task with nowadays knowledge (ultimately, one should not forget that up to now, **very little** can be proved with respect to the practical security of ciphers). A first task consists in defining the precise *security model* in which one would like to prove the security of a primitive. A rather intuitive definition of the security of a cipher is the *K-security* concept of Daemen and Rijmen [9]; obviously, this definition is extremely difficult to work with in a mathematical sense.

Definition (K-Security). *A cipher is K-secure if all possible attack strategies for it have the same expected work factor and storage requirements as for the majority of possible ciphers with the same dimensions. This must be the case for all possible modes of access for the adversary (known / chosen / adaptively chosen plaintext / ciphertext, known / chosen / adaptively chosen key relations, ...) and for any a priori key distribution.*

The NESSIE project [42] considered furthermore two additional informal (but pragmatic) security models, namely *practical security* and *historical security*: in the first model, which includes most of the cryptanalysis performed nowadays, a cipher is considered secure if the best-known attack requires too much resources by an acceptable margin. It is a very practical model as one can test the cipher with different known attacks and assess a certain security level to it. However, it is not possible to predict the security of the underlying cipher with respect to yet unknown attacks. The *historical security* of a cipher is derived according to the amount of cryptanalytic work on the ciphers performed over the years. An old cipher which resists to all cryptanalytical attacks since a long time will inevitably inspire a larger security feeling than a new cipher which has not been extensively cryptanalysed.

6.1 The security of classical ciphers

In his seminal paper about cryptography, Shannon [52] introduced a model of security known as *perfect security* or *unconditional security*. In this very strong model, one assumes that the adversary is infinitely powerful, but restricted to a ciphertext-only attack.

Let M be a plaintext, K will be a key and $Y = E(M, K)$ will be a cryptogram. C. Shannon formulated a property of a perfect security as

$$H(M) = H(X|Y)$$

and showed that the following inequality should be performed for a perfect cipher:

$$H(K) \geq H(M).$$

In this model it is supposed that a cryptanalyst has a possibility to observe a cryptotext X.

For practical cipher resistance it's required that the inequality

$$H(K|Y) \geq \alpha, \qquad H(M|Y) \geq \alpha.$$

is performed for values of the conditional entropies $H(K|Y)$ $H(M|Y)$, Nowadays for practical cipher resistance it's considered that the value

$$\alpha > 80.$$

6.2 Security of two-channel ciphers

We shall consider the cipher described in 1.2. Its usage modes are enumerated in part 2.1.

1. The key K is transmitted via a secure channel, and the outputs $Y_{\mathrm{DT}}, Y_{\mathrm{D}}$ – via an open channel. As a cryptanalyst has a possibility to observe both the output Y_{DT} and Y_{D}, this case is similar to that are of a classical symmetric system considered by Shannon.

2. The key K and the output Y_{DT} are transmitted via a secure channel, and the output Y_{D} – via an open channel. In this case an adversary cryptanalyst observes only a part of a cryptotext, and it's obvious that in this case for perfect security it's necessary to perform the equality

   $$H(M) = H(M|Y_{\mathrm{D}}).$$

   And for practical security the necessary condition is

   $$H(M|Y_{\mathrm{D}} \geq \alpha.)$$

3. The output Y_{DT} is transmitted via a secure channel, and the key K and the output Y_{D} – via an open channel. In

this case an adversary cryptanalyst observes only a part of a cryptotext and the key K, therefore, it's evident, that in this case for perfect security it's necessary to perform the equality

$$H(M) = H(M|Y_{\mathrm{D}}K).$$

And for practical security the necessary condition is

$$H(M|Y_{\mathrm{D}}K \geq \alpha.)$$

4. The key K and the output Y_{D} are transmitted via a secure channel, and the output Y_{DT} – via an open channel. In this case an adversary cryptanalyst observes only a part of a ctyptotext, and in this case for perfect security it's necessary to perform the equality

$$H(M) = H(M|Y_{\mathrm{DT}}).$$

And for practical security the necessary condition is

$$H(M|Y_{\mathrm{DT}} \geq \alpha.)$$

5. The output Y_{D} is transmitted via a secure channel, and the key K and the output Y_{DT} – via an open channel. In this case an adversary cryptanalyst observes only a part of a ctyptotext and the key K, therefore, in this case for perfect security it's necessary to perform the equality

$$H(M) = H(M|Y_{\mathrm{DT}}K).$$

And for practical security the necessary condition is

$$H(M|Y_{\mathrm{DT}}K \geq \alpha.)$$

6. The key K and a plaintext M are transmitted via a secure channel, and the outputs $Y_{\mathrm{DT}}, Y_{\mathrm{D}}$ – via an open channel. This case is similar to that one of a classical symmetric system considered by Shannon.

7. The key K and the output Y_{DT} are transmitted via a secure channel, and the output Y_{D} and a plaintext M – via an open channel. In this case the condition of perfect security can be formulated in form of

$$H(K) = H(K|Y_{\mathrm{D}}M),$$

   and condition of practical security

$$H(K) = H(K|Y_{\mathrm{D}}M) \geq \alpha.$$

   If a cryptanalyst solves the problem of defining the output Y_{DT} only, then for perfect security it's necessary to perform the equality

$$H(Y_{\mathrm{DT}}) = H(Y_{\mathrm{DT}}|Y_{\mathrm{D}}M),$$

   and for the practical one –

$$H(Y_{\mathrm{DT}}|Y_{\mathrm{D}}M) \geq \alpha.$$

8. The output Y_{DT} and a plaintext M are transmitted via a secure channel, and the key K and the output Y_{D} – via an open channel. In this case the condition of perfect security can be formulated in form of

$$H(M) = H(M|Y_{\mathrm{D}}K),$$

   and condition of practical security

$$H(M) = H(M|Y_{\mathrm{D}}K) \geq \alpha.$$

If a cryptanalyst solves the problem of defining the output Y_{DT} only, then for perfect security it's necessary to perform the equality

$$H(Y_{\mathrm{DT}}) = H(Y_{\mathrm{DT}}|Y_{\mathrm{D}}K),$$

and for the practical one –

$$H(Y_{\mathrm{DT}}|Y_{\mathrm{D}}K) \geq \alpha.$$

9. The key K and the output Y_{D} are transmitted via a secure channel, and the output Y_{DT} and a plaintext M – via an open channel. In this case the condition of perfect security can be formulated in form

$$H(K) = H(K|Y_{\mathrm{DT}}M),$$

and condition of practical security

$$H(K) = H(K|Y_{\mathrm{DT}}M) \geq \alpha.$$

If a cryptanalyst solves the problem of defining the output Y_{D} only, then for perfect security it's necessary to perform the equality

$$H(Y_{\mathrm{D}}) = H(Y_{\mathrm{D}}|Y_{\mathrm{DT}}M),$$

and for the practical one –

$$H(Y_{\mathrm{D}}|Y_{\mathrm{DT}}M) \geq \alpha.$$

10. The output Y_{D} and a plaintext M are transmitted via a secure channel, and the key K and the output Y_{DT} – via an open channel. In this case the condition of perfect security can be formulated in form

$$H(M) = H(M|Y_{\mathrm{DT}}K),$$

and condition of practical security

$$H(M) = H(M|Y_{\mathrm{DT}}K) \geq \alpha.$$

If a cryptanalyst solves the problem of defining the output Y_{D} only, then for perfect security it's necessary to perform the equality

$$H(Y_{\mathrm{D}}) = H(Y_{\mathrm{D}}|Y_{\mathrm{DT}}K),$$

and for the practical one –

$$H(Y_{\mathrm{D}}|Y_{\mathrm{DT}}K) \geq \alpha.$$

6.3 General information ratio for two-channel systems

Let M be a random message from a set of messages $\mathcal{M}$, K will be a key which is chosen at random and uniformly from the set $\mathcal{K}$, and Y_{DT} and Y_{D} will be outputs according to the system (1.2).

Entropies of inputs, outputs and keys of two-channel systems formed in accordance with the system (1.2) are interconnected by the following general information ratios:

$$\begin{aligned} &H(M, Y_{\mathrm{DT}}, Y_{\mathrm{D}}, K) = \\ &\qquad = H(MK) + H(Y_{\mathrm{DT}}Y_{\mathrm{D}}|MK) = \\ &\qquad = H(M) + H(K) = \\ &\qquad = H(Y_{\mathrm{DT}}Y_{\mathrm{D}}) + H(KM|Y_{\mathrm{DT}}Y_{\mathrm{D}}) = \\ &\qquad = H(Y_{\mathrm{DT}}Y_{\mathrm{D}}) + H(K|Y_{\mathrm{DT}}Y_{\mathrm{D}}). \end{aligned} \tag{6.1}$$

From where we get

$$H(Y_{\mathrm{DT}}Y_{\mathrm{D}}) + H(K|Y_{\mathrm{DT}}Y_{\mathrm{D}}) = H(M) + H(K) \tag{6.2}$$

For the joint entropy Y_{DT}, Y_{D} and K we have

$$\begin{aligned} H(Y_{\mathrm{DT}}Y_{\mathrm{D}}K) &= H(Y_{\mathrm{DT}}Y_{\mathrm{D}}) + H(K|Y_{\mathrm{DT}}Y_{\mathrm{D}}) = \\ &= H(Y_{\mathrm{DT}}) + H(Y_{\mathrm{D}}|Y_{\mathrm{DT}}) + H(K|Y_{\mathrm{DT}}Y_{\mathrm{D}}) = \\ &= H(Y_{\mathrm{D}}) + H(Y_{\mathrm{DT}}|Y_{\mathrm{D}}) + H(K|Y_{\mathrm{DT}}Y_{\mathrm{D}}) = \\ &= H(K) + H(Y_{\mathrm{DT}}Y_{\mathrm{D}}|K) = \\ &= [\text{ from (6.2)}] = H(M) + H(K). \end{aligned} \tag{6.3}$$

Then

$$H(Y_{\mathrm{DT}}Y_{\mathrm{D}}|K) = H(M). \tag{6.4}$$

For joint entropy of the input M, the output Y_{DT} and the key K Y_{D} the following is performed

$$\begin{aligned} H(MY_{\mathrm{DT}}K) &= H(Y_{\mathrm{DT}}) + H(MK|Y_{\mathrm{DT}}) = \\ &= H(Y_{\mathrm{DT}}) + H(M|Y_{\mathrm{DT}}) + H(K|Y_{\mathrm{DT}}M) = \\ &= H(Y_{\mathrm{DT}}) + H(K|Y_{\mathrm{DT}}) + H(M|Y_{\mathrm{DT}}K) = \\ &= H(M) + H(Y_{\mathrm{DT}}K|M) = \\ &= H(M) + H(Y_{\mathrm{DT}}|M) + H(K|Y_{\mathrm{DT}}M) = \\ &= H(M) + H(K); \end{aligned} \tag{6.5}$$

For joint entropy of the input M, the output Y_{D} and the key K the following is performed

$$\begin{aligned} H(MY_{\mathrm{D}}K) &= H(Y_{\mathrm{D}}) + H(MK|Y_{\mathrm{D}}) = \\ &= H(Y_{\mathrm{D}}) + H(M|Y_{\mathrm{D}}) + H(K|Y_{\mathrm{D}}M) = \\ &= H(Y_{\mathrm{D}}) + H(K|Y_{\mathrm{D}}) + H(M|Y_{\mathrm{D}}K) = \\ &= H(M) + H(Y_{\mathrm{D}}K|M) = \\ &= H(M) + H(Y_{\mathrm{D}}|M) + H(K|Y_{\mathrm{D}}M) = \\ &= H(M) + H(K). \end{aligned} \tag{6.6}$$

6.4 Analysis of the general scheme security at unknown flag output

If the flag output is unknown, the following problems are typical for an attacker:

1. An attacker knows the output Y_{DT}.

   1.a) Find the plaintext M;

   1.b) Find the key K;

   1.c) Find the output Y_{D};

   1.d) Find the plaintext M and the key K;

   1.e) Find the plaintext M and the output Y_{D};

   1.f) Find the key K and the output Y_{D};

   1.g) Find the plaintext M, the key K and the output Y_{D}.

2. An attacker knows the output Y_{DT} and the key K.

   2.a) Find the plaintext M;

   2.b) Find the output Y_{D};

   2.c) Find the plaintext M and the output Y_{D};

3. An attacker knows the output Y_{DT} and the corresponding plaintext M.

   3.a) Find the key K;

   3.b) Find the output Y_{D};

   3.c) Find the key K and the output Y_{D};

4. An attacker knows the output Y_{DT}, the key K and the plaintext M.

4.a) He needs to determine the output Y_{D}.

Note, that there's no need to consider all problems, because if an attacker can't solve a problem where one of the components needs to be determined, he can't solve a corresponding problem where several components need to be determined.

The problem 1.a, is not more difficult than the problems 1.d, 1.e, 1.g.

The problem 1.b, is not more difficult than the problems 1.d, 1.f, 1.g.

The problem 1.c, is not more difficult than the problems 1.e, 1.f, 1.g.

The problem 2.a, is not more difficult than the problem 2.c.

The problem 2.b, is not more difficult than the problem 2.c.

The problem 3.a, is not more difficult than the problem 3.c.

The problem 3.b, is not more difficult than the problem 3.c.

As $H(M|Y_{\mathrm{DT}}) \geq H(M|Y_{\mathrm{DT}}K)$, then the problem 1.a is not easier than the problem 2.a.

Due to $H(K|Y_{\mathrm{DT}}) \geq H(K|Y_{\mathrm{DT}}M)$, the problem 1.b is not easier than the problem 3.a.

Similarly, due to $H(Y_{\mathrm{D}}|Y_{\mathrm{DT}}) \leq H(Y_{\mathrm{D}}|Y_{\mathrm{DT}}K)$ the problem 1.c is not easier than the problem 2.b, and due to $H(Y_{\mathrm{D}}|Y_{\mathrm{DT}}) \leq H(Y_{\mathrm{D}}|Y_{\mathrm{DT}}M)$ the problem 1.c is not easier than the problem 3.b.

As the following ratios are performed

$$H(Y_{\mathrm{D}}|Y_{\mathrm{DT}}K) \geq H(Y_{\mathrm{D}}|Y_{\mathrm{DT}}KM),$$
$$H(Y_{\mathrm{D}}|Y_{\mathrm{DT}}M) \geq H(Y_{\mathrm{D}}|Y_{\mathrm{DT}}KM),$$

then the problems 2.b and 3.b are not easier than the problem 4.a.

We shall evaluate complexity of solving the problems put in front of an attacker.

To do that we shall assume that a round permutation is fixed. In this case a set of texts giving the same remainder at every transformation round is determined by the used substitution transformation $T \in \mathcal{F}_n^r$. only.

Problem 4.a.

Let us know the plaintext M, the key K and the remainder output Y_{DT}. If at each round a substitution transformation is chosen according to the deterministic rule, then an attacker will definitely determine the flags, i.e. $H(Y_{\mathrm{D}}|Y_{\mathrm{DT}}KM) = 0$.

We shall consider a case when a round substitution transformation is randomly chosen from an known key. In this case an attacker can evaluate the number of performed rounds m by the formula (4.3). For this case from the ratio (6.3) we have $H(Y_{\mathrm{D}}|Y_{\mathrm{DT}}) = m \cdot \log(k) - H(Y_{\mathrm{DT}})$. If the length of the output Y_{DT} is large enough, then in the general case an attacker has, k^m variants of pairs $(Y_{\mathrm{DT}}, Y_{\mathrm{D}})$ and with probability close to 1, there's the only pair with the set core among the variants. Note that with increase of m, entropy $H(Y_{\mathrm{DT}})$ decreases. Therefore, required security of the cipher can be obtained thanks to increasing the number of performed rounds.

Problem 2.b.

Let the key K and the remainder output Y_{DT} be known. If at each round a substitution transformation is chosen by the deterministic rule, then from the ratio (6.3) we have

$$H(Y_{\mathrm{D}}|Y_{\mathrm{DT}}) = H(M) - H(Y_{\mathrm{DT}}).$$

If a round substitution transformation is randomly chosen from a known key, then the conditional entropy

$$H(Y_{\mathrm{D}}|Y_{\mathrm{DT}}) = m \cdot \log(k) + H(M) - H(Y_{\mathrm{DT}}).$$

Consequently, the required security of the cipher can be obtained thanks to increasing the number of performed rounds.

Problem 2.a.

Let the key K and the remainder output Y_{DT} be known. If the plaintext M is unknown and the number of performed rounds m is unknown, then for the known core Y_{DT} even at known keys K there's an infinite set of texts at which you can get such a core. At the limited number of rounds a set of texts corresponding to the given core is finite. Further we shall assume that an attacker knows the number of performed rounds m.

We shall evaluate the number of texts having the same remainder.

In the considered scheme an MV2-transformation is performed at each round. As it is shown in 3.2, for the known core C of the length $|C| = l$ bits, the number of its preimages for one round is determined by the expression (3.33).

As whitening is used at the scheme input, then, according to the theorem 3.1, the remainder output can be considered as a uniform sequence. Therefore, for evaluating the number of preimages N_m of the core C after performing m rounds of the scheme one can use the inequality (3.49). Due to this inequality we have:

$$\log N_m > \frac{1}{\mathrm{K_c}}\frac{|C|}{n} + \ldots + \frac{1}{\mathrm{K_c}^m}\frac{|C|}{n},$$

From where we have:

$$N_m \geq 2^{\frac{1-\mathrm{K_c}^m}{\mathrm{K_f}} \cdot \frac{|C|}{n}}. \tag{6.7}$$

Accordingly, if L is the number of symbols in the plaintext which went to the scheme input is known, then to evaluate N_L is the number of texts corresponding to the known core from the inequality (6.7) we have

$$N_L \geq 2^{\frac{1-\mathrm{K_c}^m}{\mathrm{K_f}}\cdot L}. \tag{6.8}$$

Thus, complexity of recovering the plaintext at known keys can be evaluated as

$$H(M|Y_{\mathrm{DT}}K) = O(\frac{1-\mathrm{K_c}^m}{\mathrm{K_f}}\cdot L).$$

Problem 3.a.

Let the plaintext M and the remainder output Y_{DT} are known. In this case from the ratios (6.5) we have:

$$H(K|Y_{\mathrm{DT}}M) = H(K) - H(Y_{\mathrm{DT}}).$$

If each transformation T_i from the key K is at random and uniformly chosen from a set of the transformations $\mathfrak{F}_n^r$, then entropy $H(K) = k\cdot\log(2^n!)$, using the Stirling formula we have $H(K) \approx k\cdot((n-1.443)\cdot 2^n + \frac{n+1}{2} + 0.826) \geq k\cdot(n-2)\cdot 2^n$. In this case, for secure use at $n \geq 8$ it's enough to require performing such the number of rounds that for the core length the following would perform $|Y_{\mathrm{DT}}| \leq (k-1)(n-3)\cdot 2^n$ bits.

If a round substitution tramsformation is chosen at random from the key K, then for each m-round encryption of a message M entropy of the real key K^* will equal $H(K^*) = H(K) + m\cdot\log k$.

6.5 Analysis of the general scheme security at unknown core output

If the core output is unknown, then the following problems are typical for an attacker:

5. An attacker knows the output Y_{D}.

   5.a) Find the plaintext M;

   5.b) Find the key K;

   5.c) Find the output Y_{DT};

   5.d) Find the plaintext M and the key K;

   5.e) Find the plaintext M and the output Y_{DT};

   5.f) Find the key K and the output Y_{DT};

   5.g) Find the plaintext M, the key K and the output Y_{DT}.

6. An attacker knows the output Y_{D} and the key K.

   6.a) Find the plaintext M;

   6.b) Find the output Y_{DT};

   6.c) Find the plaintext M and the output Y_{D};

7. An attacker knows the output Y_{D} and the corresponding plaintext M.

   7.a) Find the key K;

   7.b) Find the output Y_{DT};

   7.c) Find the key K and the output Y_{DT};

8. An attacker knows the output Y_{D} the key K and the plaintext M.

   8.a) He needs to determine the output Y_{DT}.

Note, that there's no need to consider all problems, because if an attacker can't solve a problem where one of the components needs to be determined, he can't solve a corresponding problem where several components need to be determined.

The problem 5.a, is not more difficult than the problems 5.d, 5.e, 5.g.

The problem 5.b, is not more difficult than the problems 5.d, 5.f, 5.g.

The problem 5.c, is not more difficult than the problems 5.e, 5.f, 5.g.

The problem 6.a, is not more difficult than the problem 6.c.

The problem 6.b, is not more difficult than the problem 6.c.

The problem 7.a, is not more difficult than the problem 7.c.

The problem 7.b, is not more difficult than the problem 7.c.

As whitening of the plaintext by a stream cipher takes place before performing main transformation rounds, one can assume that the plaintext consists of symbols that are chosen equiprobably from $\{0, 1\}^n$. In this case at analysis of the flags output $H(M) = n \cdot L$. Then after performing the first round in the output we have the flag F_1 and the remainder C_1, for which $H(F_1) = \mathrm{K_f} \cdot n \cdot L$, and $H(C_1) \geq \mathrm{K_c} \cdot n \cdot L$. Due to the theorem about distributing bit remainder, an obtained remainder looks like a uniform text. And due to the statement about mathematical expectation of the remainder length $|C_1| = \mathrm{K_c} \cdot n \cdot L$. From where

$$H(Y_\mathrm{D}) \geq n \cdot L \cdot (1 - \mathrm{K_c}^m). \tag{6.9}$$

Note that if in the general scheme of the cipher a substitution transformation is chosen at random from the fixed key K, then, there are k^m variants of the real key at performing m rounds of the algorithm for the set key K. Consequently, at equiprobqble choice of variants for entropy of the real key used for the encryption K_{in} the following is performed

$$H(K_{in}) = H(K) + m \cdot \log k. \tag{6.10}$$

To determine complexity of these problems we shall use general information dependencies (6.2) – (6.4), (6.6).

From (6.3) we have:

$$H(Y_\mathrm{D}) + H(Y_\mathrm{DT}|Y_\mathrm{D}) + H(K|Y_\mathrm{DT}Y_\mathrm{D}) = H(M) + H(K). \tag{6.11}$$

From (6.6) we have:

$$H(Y_\mathrm{D}) + H(M|Y_\mathrm{D}) + H(K|Y_\mathrm{D}M) = H(M) + H(K). \tag{6.12}$$

Problem 8.a.

Let the plaintext M, the key K and the remainder output Y_D be known. If at each round a substitution transformation is chosen by the deterministic rule, then an attacker will exactly determine the core.

Let a round substitution transformation be chosen from a known key. If a cryptanalyst knows the flags output Y_D, the key K and the plaintext M, then he can exactly determine the remainder output. Indeed, the key K consists of k MV2-transformations $T_i = (c_i, f_i)$, at that $k \ll \#\mathfrak{F}$, therefore, the probability that among flags transformations there will be at least 2 identical of them is

$$1 - \frac{\#\mathfrak{F} \cdot (\#\mathfrak{F} - 1) \cdot \ldots \cdot (\#\mathfrak{F} - s + 1)}{(\#\mathfrak{F})^s} \approx 0,$$

where $\#\mathfrak{F}$ can be found by the formula (3.3).

Then comparing M and the first flags one can determine the transformation T_{i_1} which was used to obtain them, and consequently, get the remainder of the first round. A remainder of the first round is an input text of the second round, therefore, comparing it with the second flags one can determine the transformation T_{i_2} which was used to obtain them, and so on... till we shall get a remainder of the last round that is the required core.

Thus, $H(Y_{\mathrm{DT}}|Y_{\mathrm{D}}KM) = 0$ and using mode 8 is not secure.

Problem 6.a. Let the key K and the flag output Y_{D} be known. It's evident, that in this case the ratio $H(M|Y_{\mathrm{D}}K) = H(Y_{\mathrm{DT}}|Y_{\mathrm{D}}K)$ is performed.

If the number of performed rounds m is known, then one may evaluate the input length M and the estimated output length $|Y_{\mathrm{DT}}|$ can be evaluated as $|Y_{\mathrm{DT}}| = \mathrm{K_c}^m \cdot |M|$.

If a round substitution transformation is chosen at random from the key K, then from the formulae (6.10) and (6.12) we have

$$H(M|Y_{\mathrm{D}}) = H(M) - H(Y_{\mathrm{D}}) + m \cdot \log k - H(K|Y_{\mathrm{D}}M).$$

From where, due to (6.12)

$$H(M|Y_{\mathrm{D}}) \geq \mathrm{K_c}^m \cdot n \cdot L + m \cdot \log k.$$

Therefore, the complexity of the problem of plaintext recovering satisfies modern requirements if

$$\mathrm{K_c}^m \geq \frac{\alpha - m \cdot \log k}{|Y_{\mathrm{D}}|}. \tag{6.13}$$

The requirement (6.13) is always performed from a pseudorandom choice of substitution transformations if $m \geq \dfrac{\alpha}{\log k}$ rounds are performed. Especially at $\alpha = 80$ for a basic implementation of the MV2 cipher the requirement is fulfilled after 16 transformation rounds irrespective of the length of an input message.

Note, that if we reject randomization of the MV2 cipher which occurs at each round due to random choice of a substitution transformation, then from (6.13) it follows, that $m \leq \dfrac{\log(n \cdot L) - 7}{\log(1/\mathrm{K_c})}$, i.e. the number of possible transformation rounds depends on the plaintext length. For instance, for the basic implementation of the cipher, in case of rejecting a pseudorandom choice of a substitution transformation at each round, for a 1MB text it's secure to perform no more than 41 round, and for a 1kB text – no more than 16 rounds, and for a 16-byte text - no more than one round.

Problem 6.b.

Let the key K and the flag output Y_{D} be known. Due to (3.51) the problems 6.b and 6.a. – are equivalent and should satisfy the same ratios for the required number of rounds.

Problem 7.a. Let the plaintext M and the flag output Y_{D} be known. From the ratio (6.1) it follows:

$$\begin{aligned} H(Y_{\mathrm{DT}} Y_{\mathrm{D}} K|M) = [\text{ so } H(Y_{\mathrm{DT}}|Y_{\mathrm{D}} K M) = 0] = \\ = H(Y_{\mathrm{D}} K|M) = H(K). \end{aligned}$$

From where $H(K|Y_{\mathrm{D}} M) = H(K) - H(Y_{\mathrm{D}}|M)$.

At a fixed choice of a round substitution transformation from the key K entropy $H(K) = k \cdot \log\bigl(2^n!\bigr)$, and conditional entropy at performing m rounds doesn't exceed $H(Y_D|M) \leq m \cdot \log\bigl(\prod_{i=r}^{n-1} 2^i!\bigr)$. Therefore

$$H(K|Y_{\mathrm{D}}M) \geq \log \frac{\bigl(2^n!\bigr)^k}{\bigl(\prod_{i=r}^{n-1} 2^i!\bigr)^m}.$$

If the key is chosen at random at each round, then

$$H(K|Y_{\mathrm{D}}M) \geq k \cdot \log\bigl(2^n!\bigr) + m \cdot \log k - m \cdot \log\Bigl(\prod_{i=r}^{n-1} 2^i!\Bigr). \quad (6.14)$$

Problem 7.b. Let the plaintext M and the flag output Y_{D} be known.

From the ratio (6.1) it follows:

$$H(Y_{\mathrm{DT}}|Y_{\mathrm{D}}M) = H(K|Y_{\mathrm{D}}M).$$

Therefore the problems 7.a and 7.b have the same complexity.

6.6 Some conclusions

From the point 6.4 and 6.5 we can draw the following conclusions

- for security of the general scheme the length of the core output shouldn't be too small $|Y_{\mathrm{DT}}| \geq \alpha$;
- a pseudorandom choice of a substitution transformation from a key increases security of the scheme in case, when one of the outputs is unknown;
- using the general scheme is not secure, if an adversary cryptanalyst knows the flag output, the plaintext and the key.

Chapter 7
Basic implementation of the MV2 algorithm

7.1 Description

In this application one of the implementations of the general scheme of harming (see 4) is described. It is one of the implementations of the MV2 algorithm (see 4.3)), which we shall further call as *basic*.

This implementation has the following design features:

- a key is a short ($128 \div 2048$ bits) sequence (master key), from which 32 tables are generated that set an MV2-transformation with the parameters $r = 3$ and $n = 7$;
- a stream cipher RC4 is used for whitening;
- a 128-bit linear transformer with a high degree of diffusion is used as a permutation transformation.

We shall describe input and output parameters of the basic implementation.

Encryption

Input:	plaintext	$M\,(8 \times L$ bits)
	secret key	$K\,(128,\ 256,\ 512,$ $1024,\ 2048$ bits)
	number of rounds:	m
	or maximal core length:	L_c
Output:	ciphertext	$\Big(C\,,\,F\Big)$

Decryption

Input:	ciphertext	$\Big(C\,,\,F\Big)$
	secret key	$K\,(128,\ 256,\ 512,$ $1024,\ 2048$ bits)
Output:	error message	
	or a plaintext	$M\,(8 \times L$ bits)

To ensure the ciphertext could be decrypted back to the message, the encryption transformation has to be invertible, but it's not necessarily to use identical algorithms for encryption and decryption. In MV2, encryption and decryption is made by different algorithms. As mentioned in 4, a global structure of the encryption algorithm of the MV2 cipher may be shown as an SPN (see Fig. 4.1).

The whole encryption process made by cryptographic algorithm MV2 could be divided into some rounds, with interleaving of linear and non-linear transformations. Each round consists of a *linear layer* and *non-linear layer* . Mappings with images of various lengths are used to implement non-linear transformations. These mappings are set with secret tables which constitute key data.

The number of rounds in the basic implementation of the MV2 algorithm can be set directly or indirectly by setting the upper bound for the core length. In any case there will be no less than 16 transformation rounds performed. Besides, at a set remainder of the upper margin a number of rounds is defined automatically at reaching the set core length.

A plaintext is whitened by the stream encryption cipher RC4 before performing main rounds. One of the permutations is used as an RC4 key. This permutation sets an MV2-transformation and is generated from the key.

A round of the basic implementation

Each round consists of two transformations: a substitution transformation and a permutation transformation. The permutation transformation is made locally. The processed message is divided into 128-bit blocks, each subjected to the same transformation, which rearranges its bits. If the text length is not multiple 128, the last incomplete block is not processed. The permutation transformation is followed by the substitution one set by the selected table for this round.

The permutation transformation is the linear ensuring high degree of local diffusion and it is similar to that one described in [3]. This transformation permutes 128 bits recorded in four 32-digit words **"abcd"** in the following way:

$\mathbf{a} = \mathbf{a} \lll 13$

$\mathbf{c} = \mathbf{c} \lll 3$

$\mathbf{d} = \mathbf{d} \bigoplus \mathbf{c} \bigoplus (\mathbf{a} \ll 3)$

$\mathbf{b} = \mathbf{b} \bigoplus \mathbf{a} \bigoplus \mathbf{c}$

$\mathbf{d} = \mathbf{d} \lll 7$

$\mathbf{b} = \mathbf{b} \lll 1$

$\mathbf{a} = \mathbf{a} \bigoplus \mathbf{b} \bigoplus \mathbf{d}$

$\mathbf{c} = \mathbf{c} \bigoplus \mathbf{d} \bigoplus (\mathbf{b} \ll 7)$

$\mathbf{a} = \mathbf{a} \lll 5$

$\mathbf{c} = \mathbf{c} \lll 22$

where $\lll$ denotes rotation, $\ll$ denotes shift, and $\bigoplus$ is EXCLUSIVE OR.

At each round the byte Substitution transformation is performed by applying one of 32 key substitution transformations to each byte of the text. The transformation is selected using values of R from the GPS.

A substitution transformation is an MV2-transformation (see 3.2).

This transformation maps a n-bit string x into a pair (c, f), consisting of two variable length strings. In the basic realization $n = 8$ and $r = 3$, therefore such transformations are called *byte substitution transformations.* Each byte of the input text is mapped into a pair of bit strings, the first one (remainder) is 3 to 7 bits long, the second one is a value code in the range from 1 to 6. This transformation can be set by the table, where a permutation $x_0, \dots x_{255}$ of values 0 to 255 is in the left part and images, consisting of the "remainder" and "flag" parts, are stored in the right part (see Table 7.1 and 3.2). The substitution transformation is described in section 3.1.2.

Key

The key is an arbitrary binary string which can have a length of 128, 256, 512, 1024 or 2048 bits. In the algorithm a special key transformation is used, which maps a received key into a set of substitution tables. A set of transformations is set by a string having the following format: the first 256 bytes contain permutation of all the numbers from 0 to 255 that set a MV2-mapping, and 4 standby bytes. At using, the permutation is displayed into a table which sets an MV2-transformation, as shown in the table 7.1.

The algorithm of substitution transformation generation is resistant to a linear and differential cryptanalysis.

In the basic realization of the algorithm the number of substitution tables in the key is limited to 32 (in this case to

Table 7.1: *Task of a substitution transformation*

Symbol	Image length	Remainder	Flag
s_1	3	000	00001
s_2	3	001	00001
...	...	...	...
s_8	3	111	00001
s_9	4	0000	0001
...	...	...	...
s_{24}	4	1111	0001
s_{25}	5	00000	001
...	...	...	...
s_{56}	5	11111	001
s_{57}	6	000000	01
...	...	...	...
s_{120}	6	111111	01
s_{121}	7	000000	1
...	...	...	...
s_{248}	7	1111111	1
s_{249}	3	000	00000
...	...	...	...
s_{256}	3	111	00000

give a transformation number 5 binary bits are required). In versions for specific applications this number of tables may be increased.

Random number generator

The basic implementation of the algorithm is an iterative probabilistic cipher.

The generator is used to randomize the cipher, i.e. a random number of a table is generated and used at each current encryption round. In the basic version we use a random number generator (RNG) built on the basis of the affine transformation:

$$x_{t+1} = 2^{13}(x_t + x_{t-1} + x_{t-2}) \bmod (2^{32} - 5).$$

The initial state of the RNG is reset by the timer during initialization the device.

The nature of the generator is of no importance for the cipher, therefore any, also a physical, RNG may be used.

Presentation of output data

Algorithm's output

Output data consist of two binary sequences we call a core and string of flags. The core is a remainder obtained at the last round. A String of flags is a concatenation of output flags obtained at all transformation rounds. For convenience of decryption, the flags round outputs are put in a reverse order (the last flag round output, the one before, ..., the first flag round output) without separators.

Output of each round

As substitution transformations have bit outputs, while the main data storage unit in modern computer systems is byte, in the general case, the algorithm's outputs have to be complemented to a byte value. At the same time we have to know the true length in bits for decryption. Therefore, 1 byte of service data is fore-added to the obtained remainder, this byte comprises the table number (5 bits) and the number of real bits in the last byte (3 bits).

For such a presentation, strings of flags practically have no redundancy.

7.2 Key schedule of the cipher MV2

Our strategy in selecting S-boxes is to choose them completely at random. While this approach can be insecure for small S-boxes (for instance, Biham and Shamir noticed [5] that replacing S-boxes of DES by random S-boxes yielded ciphers that were far weaker towards differential cryptanalysis than the original algorithm), theoretical works by O'Connor [44]-[47] and later by Youssef and Tavares [54]-[56] have shown that large random S-boxes are in average very resistant to linear and differential cryptanalysis. A way to choose random S-boxes is to make them key-dependent. As example, this approach has been used in the algorithms IDEA, RC5, RC6, Blowfish or Twofish.

7.2.1 Notations

The following is a list of notations and glossary terms:

- **word** : a 32 bit quantity.
- **master key** : the user defined key, or initial secret key.
- N : a bit length of a master key.
- $x \lll n$: denote a cyclic left shift or cyclic rotation of a word x by n bits to the left.

In the calculation, g represents a bit-wise operation defined by

$$g(x2, x1, x0) = x2x1 \oplus x1x0 \oplus x0x2, \tag{7.1}$$

where each x_i is a word, $x_i x_j$ represents the bit-wise AND, while $A \oplus B$ the bit-wise XOR of the two data involved.

There are many different criteria which can be applied to key schedule analysis, including : weak keys, semi-weak keys, complementation properties, reconstruction of master key or other subkeys from recovered subkey bits, dependence of subkeys on the (full/partial) master keys material (key diffusion). These aspects will be taken into account in the design and analysis the key schedule algorithm of MV2.

Design principle and criteria:

- maximize avalanche in the subkeys,
- reconstruction of master key or other subkeys from recovered subkey bits,
- dependence of subkeys on the (full/partial) master keys material (key diffusion).

The key schedule MV2 is defined for master keys of length $N = 128$, $N = 256$, $N = 512$, $N = 1024$ and $N = 2048$ bits.

7.2.2 Properties of Random S-boxes

Since the object of this proposal is a new cipher using key-dependent S-boxes, it will be useful to investigate the average properties of random invertible $n \times n$ S-boxes. A series of combinatorial results have demonstrated that a randomly chosen s-box of sufficient size will possess several of these desirable properties with high probability.

Completeness

O'Connor proved in [46] that a randomly chosen $n \times n$ invertible S-box has a high probability of being complete for sufficiently large n. In fact, he showed that the probability that such an S-box is not complete is

$$o\left(\frac{\sqrt{2^n}}{2^{2^{n-1}+n+1}}\right).$$

For an exact formula, see [44].

Avalanche and Strict Avalanche

The authors have not found any results giving the probability that a random invertible $n \times n$ S-box satisfies AVAL or SAC (although there are bounds on the probability that a random function $f : \{0,1\}^t \to \{0,1\}$ satisfies SAC). On the other hand, a number of theoretical and experimental results exist concerning the AVAL property for SPNs. Heys and Tavares developed a probabilistic model for the AVAL property of an SPN [21]. Their results for $N = 64$ and $n = M = 8$, using randomly selected S-boxes and the fixed permutation of Kam and Davida [23], indicate that AVAL is reasonably satisfied after 5 or more rounds. In fact, if E_R is the expected number of output bit changes after R rounds when one input bit is flipped, and we define $\varepsilon = |1 - E_R/(N/2)|$, then $\varepsilon \leq 10^{-5}$ for $R \geq 7$.

Nonlinearity

For an invertible $n \times n$ S-box S and an integer $2L(0 \leq L \leq 2^{n-2})$, Youssef and Tavares [56] prove that

$$\mathbf{prob}[nl(S) \leq (2^{n-1} - 2L)] < \\ < \frac{2(2^{n-1}!)^2(2^n-1)^2}{2^n!} \sum_{i=L}^{2^{n-2}} \binom{2^{n-1}}{2^{n-2}+i}^2.$$

If $n = 8$, we have, for example, $\mathbf{prob}[nl(S) \leq 64] < 1.4 \cdot 10^{-11}$ and $\mathbf{prob}[nl(S) \leq 80] < 4.6 \cdot 10^{-5}$. Experimental results support the theoretical result of (3). For example, Heys [20] generated 200 random invertible $n \times n$ S-boxes and found that each satisfied $86 \leq nl(S) \leq 98$.

XOR Table Distribution

If S is a randomly chosen invertible $n \times n$ S-box, and $0 \leq A \leq 2^n$ is an even integer, a formula of Youssef and Tavares gives the probability that the maximum XOR table entry of S (denoted maxXOR(S)) is $\geq A$ [56]. For example, if $n = 8$ and $A = 16$, we have

$$\mathbf{prob}[\text{maxXOR}(S) \geq 16] \leq 0.0042.$$

Cyclic Properties

There is some indication that the cyclic properties (cycle length, number of cycles) of an S-box are related to other cryptographic properties. Youssef et al. give experimental results which show that, on average, S-boxes with fewer fixed points have higher nonlinearity and lower maximum XOR table entries [57]. They prove that the expected number of fixed points for a random invertible S-box is 1, with a variance of 1. They also state that the expected value and variance of the number of cycles is approximately $\ln 2^n \approx 0.69n$; and that the expected cycle length is $2^{n-1} + 1/2$, where the expected cycle length is defined as the value of the length of the cycle to which a randomly chosen element belongs.

7.2.3 Random S-Box Generation Process

The fact that the S-boxes are unknown to the cryptanalyst is one of the principal strengths of our system, since both linear and differential cryptanalysis require known S-boxes. It is not apparent that the pseudo-random nature of the S-boxes introduces any exploitable weakness into the system. The results about properties of random S-boxes indicate that if the S-boxes are generated from the key in a sufficiently random fashion, each S-box has a high probability of being complete, possessing fairly high nonlinearity, and having its largest XOR table entry < 16.

The key, K, is used to generate a series of subkeys $KX_1, KX_2, \ldots, KX_{MR}$, by application of a key expansion function f. A second function g generates the i-th S-box, S_i, from some subset of subkeys KX_i. The functions f and g must meet certain requirements. First of all, they must produce S-boxes which are satisfactorily random, in order to achieve the results of Section 5.2. Secondly, even though the S-boxes are in principle secret, we want the generation process to be such that if a cryptanalyst determines one of the S-boxes, this does not yield any information about any other S-box. There are a number of ways to achieve this security. One approach is to make f cryptographically secure; g can then be any simple function which generates pseudo-random S-boxes from the KX_i. Another method is to make g one-way, for example a secure hash function, or a many-to-one function. Then it is only required that f be sufficiently random.

Our algorithm chooses an S-box uniformly from the set of all invertible 8×8 S-boxes. It is based on the design concept of ”mixing operations from different algebraic groups”.

We are also considering a number of other options for f and g. As this work progresses, test results for these different methods will enable us to determine the approach which yields the most secure SPN. Note that we plan to keep the S-box generation process separate from the SPN itself. Two other cryptosystems Khufu and Blowfish also use key-dependent S-boxes, but each incorporates the cryptosystem itself in some initial state to pseudo-randomly generate the S-boxes used for actual encryption and decryption. By avoiding this self-referential approach, we hope that the analysis of our system will be simplified.

Key Expansion Functions

Each algorithm key schedule has a KX (key expansion) table cosisting of 256 words of 32 bits, pseudo-randomly generated from the master key. Algorithms for creating the tables are different for each key schedule. The table may be computed when a master key is setup. The algorithms use a masked table T cosisting of 8 pseudo-random words of 32 bits. Elements of a master key and elements of table T are input for nonlinear transformations.

The purpose of the key expansion function is to pseudo-randomize the KX array, allowing each input bit to influence every other bit.

Criteria for sequence of words $kx[0], kx[1], \ldots, kx[255]$:

- Bits of each word should be equiprobable and statistically independent from each other;
- Bits of each word should be statistically independent from bits of several next words;
- The number of possible target sequences should not be essentially less than the number of possible master keys.

S-Box Generation Function

The code, for creating S-boxes from an array KX, is the same for each algorithm key schedule. The function, creating S-boxes, has 8 internal state variables, each an unsigned 32-bit word. They are denoted $s_0, \ldots, s_7$. Before the first pass over the KX array, they are initialized with key-dependent values, which are computed by the key expansion function. All S-boxes consist of a permutation of the numbers from 0 to 255. We compute $255 \cdot 32$ pseudo-random numbers in the variable x. The mixture of different operations is used for computing a pseudo-random number x.

7.3 Statistical estimations of output data and resistance

Evaluation of output lengths

As it was mentioned it's important to know lengths of output data for practical implementation.

Let M be a plaintext of the length $L(M)$ bytes, and (C, F) be a ciphertext after m rounds of transformations.

In the basic realization of the MV2 algorithm one service byte is added to each result of a round substitution transformation. If $\mathbf{E}(L_m(C))$ is the expectation of the length of the core and $\mathbf{E}(L_m(F))$ is the expectation of the length of the string of flags then

$$\mathbf{E}\Big(L_m(C)\Big) \approx \mathrm{K_c}^m \cdot \Big(L(M)+1\Big) + \frac{159}{128} \cdot \frac{1-\mathrm{K_c}^{m+1}}{\mathrm{K_f}}, \quad (7.2)$$

$$\begin{aligned}\mathbf{E}\Big(L_m(F)\Big) \approx (1-\mathrm{K_c}^{m+1}) \cdot (L(M)+1) + \\ +\frac{225}{128} \cdot m - 1 - \frac{1-\mathrm{K_c}^{m+1}}{\mathrm{K_f}},\end{aligned} \quad (7.3)$$

where $\mathrm{K_c} = \dfrac{97}{128} = 0.7578125$ and $\mathrm{K_f} = \dfrac{31}{128} = 0.2421875$ (see expressions (3.41) and (3.42) from the section 4.1).

If the core length shouldn't exceed some length L_c, then

$$m \approx 1 + \frac{\log L_c - \log L(M)}{\log \mathrm{K_c}} \tag{7.4}$$

and the total length of the flags

$$\mathbf{E}(L_m(F)) \approx L(M) - L_c + 2 \cdot \frac{\log L_c - \log L(M)}{\log \mathrm{K_c}}. \tag{7.5}$$

The total output length:

$$\mathbf{E}(L(C) + L(F)) \approx L(M) + 2 \cdot \frac{\log L_c - \log L(M)}{\log \mathrm{K_c}}. \tag{7.6}$$

For random L-bytes input text for one round of transformation from the statement 3.3 follows that a number of bytes in the output of the remainder will be restricted to:

$$(\mathrm{K_c} - \sigma_c/8) \cdot L \ \leq\ |C|/8 \ \leq\ (\mathrm{K_c} + \sigma_c/8) \cdot L,$$

and a number of bytes in the output flags will be restricted to:

$$(\mathrm{K_f} - \sigma_f/8) \cdot L \ \leq\ |F|/8 \ \leq\ (\mathrm{K_f} + \sigma_f/n) \cdot L,$$

with the probability no less than $1 - L^{-2}$. In these inequalities σ_c and σ_f are standard deviations of the output text length from average values. For a uniformly distributed input text $\sigma_c \approx 1,2$ and $\sigma_f \approx 4,2$.

Note that the value of a standard deviation of the remainder output length is not large and is about 1 bit per plaintext symbol. At the same time the value of a standard deviation of the flag output length is more than 1/2 byte per 1 byte of a plaintext. On the other hand at each round the flag output length is defined by the length of a remainder obtained at a previous round, therefore with the probability $1 - L^{-2}$ after executing the m-th round the remainder length $L_C^{(m)}$ (in bytes) won't exceed the value

$$L_c^{(m)} \leq (\mathrm{K_c} + \sigma_c/8)^m \cdot L + 10,$$

and the flag output length $L_F^{(m)}$ (in bytes) won't exceed the value

$$L_F^{(m)} \leq \frac{3}{4}(\mathrm{K_c} + \sigma_c/8)^m \cdot L + 10.$$

About resistance

Note that a simple attack meet-in-the-middle cannot be implemented for MV2. We did not find a better attack on MV2 with 16 rounds other than a brute force attack. There are probably faster attacks, but they should require an unreal amount of selected open texts and memory volume.

Unlike other cryptographic transformations, in our case we may consider not only variants of unknown keys, but other variants as well – an unknown core or an unknown string of flags at known or unknown keys.

Here the plaintext M shall be considered as a uniform sequence of symbols $x \in \{0,1\}^n$. Such consideration is justified as, from one side, in the basic realization before implementation of main rounds, the plaintext is being processed by a stream cipher (noise), from the other side, as the result of a certain number of rounds, a randomized text goes to the input of a byte substitution transformation.

Note that for the MV2 cipher the complexity of attacks grows together with the length of the plaintext.

Evaluation of number of texts having a set remainder and unknown flags

If only a core is known, then it is not possible to recover the plaintext even if the keys are available.

As a round permutation is fixed, a set of texts having the same remainder is determined by the substitution transfortmation $T = \Big(c, f\Big) \in \mathfrak{F}_8^3$. If L_c is a number of bytes in the output of the core and m is a number of performed rounds, then N_C is a number of possible plaintexts corresponding to the given core will be no less than see the formula (4.4):

$$N_C \geq 2^{\left(\frac{128}{31}\left(\frac{128}{97}\right)^m - \frac{1}{31}\right)\cdot L_c}$$

For example, if the known core has the length of 1032 bits (128 bytes + 1 byte of service information), $n = 8$, $r = 3$ and $m = 10$ rounds was executed, then no less than 2^{67657} variants of a plaintext (if the key is known) is possible.

Evaluations of number of texts having required flags and an unknown remainder

If the core is unknown, the plaintext's length and, possibly, the number of rounds is unknown as well, this sharply decreases probability to select the plaintext.

Assume in the result of encryption of the plaintext M by the MV2 cipher with the key K, we obtained a cryptotext $(C, F) = MV2(K, M)$. As the cipher MV2 is a pseudorandom function, the task of finding M using known K and F is $2^{H(C)}$ hard. Note that in the real applications where file sizes are bigger than 1024 bytes, after encryption we get $|C| > 128$ bits. Hence, $H(C) > 128$, therefore complexity of finding X by the known K and F corresponds to modern requirements.

If the number of rounds m is known, then number N_F of plintexts which have the string of flags F, is

$$N_F \approx 2^{\frac{\mathrm{K_c}^m}{1-\mathrm{K_c}^{m+1}}\cdot|F|},$$

where $|F|$ is a length of the string of flags (bits) and $\mathrm{K_c} = \frac{97}{128}$.

About inherited properties of the plaintext

If a cryptoanalyst has no a single pair "message-cryptogram", the only thing he might use, would be analysis of properties of the open text that are being inherited by cryptograms. I.e., the real plaintext is replaced by its model, reflecting its most important properties. Then the cryptoanalysis may be built, for example, on the statistical solutions theory. During such an approach, the most important features of the plaintext model are its frequency characteristics. It's practically not possible to reveal a correlation between frequency characteristics of the chosen plaintext model and those ones of flags due to the following:

There are 6 different digits in the alphabet of flags. For each key mapping $T = (c, f) \in \mathfrak{F}_8^3$, there are 8 images with flags 5 and 6, and 2^{8-j} images with the flag j, $1 \leq j \leq 4$. Therefore, if the language model have n symbols, then for the random mapping T, expectations of the numbers t_j symbols having an image with the flag j are $\mathbf{E}(t_1) = n/2$, $\mathbf{E}(t_2) = n/4$, $\mathbf{E}(t_3) = n/8$, $\mathbf{E}(t_4) = n/16$, $\mathbf{E}(t_5) = n/32$, $\mathbf{E}(t_6) = n/32$. I.e., practically all the symbols cannot be identified using the first flag output. So the frequency characteristics of the first flag output does not correlate with the frequency characteristics of the model language.

Evaluations of the number of texts at an unknown key

If an unknown transformation $T = (c, f) \in \mathfrak{F}_8^3$ is applied to the plaintext $M = x_1 \| \dots \| x_L$ of the length L and we have available the core — $c(M) = c(m_1) \| c(m_2) \dots \| c(m_L)$ and the string of flags — $f(M) = f(m_1) \| f(m_2) \dots \| f(m_L)$.

If the plaintext M is unknown and L is big enough ($L > 2^9 = 512$), then there are $2^n!$ possible transformations T and, correspondingly, $2^8! \approx 2^{1684}$ possible variants of the plaintext.

In the case of the uniformly distributed input at each transformation round, there are $\prod_{i=r}^{n-1} 2^i!$ different output texts $c_i(M_{i-1})$. At $n = 8$ and $r = 3$ this makes about 2^{491}. Accordingly, there are about 2^{1193} images $c_i(M_{i-1})$

Security of the cascade

Security of a cascade cipher is characterized by the following theorem ([33]):

Theorem 7.1 *A cascade cipher has at least the same security as the first cipher in the cascade.*

A basic implementation of the MV2 algorithms is a cascade of ciphers. In such system, as in any other, the plaintext M and the secret key K are random values. Statistics of M depends on the nature of the source of plaintexts, while the statistics of K is controlled by the cryptographer. In usual ciphers the encryption process is deterministic, i.e., the cryptotext Y is uniquely determined by the plaintext M and the key K.

A cipher has *property of non-expanding* if there is an ascending sequence of positive integer $n_1, n_2, \ldots$, such, that the first n_i digits $Y_1, Y_2, ...Y_{n_i}$ of the cryptotext together with the secret key uniquely determine the first n_i digits $X_1, X_2, ...X_{n_i}$ of the open text for $i = 1, 2, \ldots$. Ciphers with the property of non-expanding are called non-expanding. A single-round MV2 cipher is a non-expanding cipher, we may simply assume $n_i = \sum\limits_{j=1}^{i} f_i$, where $f_1, f_2, ...$ are flags, and $Y_1, Y_2, ...Y_{n_i}$ are substrings of the remainder, such, that $|Y_i| = f_i$. The following fact is well known (see. [30]).

Property "random input – random output" for non-expanding ciphers: For each selection of k of the secret key K, the cascade consisting of binary-symmetric source (BSS) and a non-expanding cipher creates a BSS as well. Moreover, for any probabilistic key distribution, this cascade shall generate a cryptotext sequence $Y_1, Y_2, ...$ which is statistically independent from the secret key K.

As defined by Shanon, a cipher is *perfectly secure* If the key K is statistically independent from the cryptotext sequence $Y_1, Y_2, ...$ At any attacks using a known ciphertext against a perfectly secure cipher, the attacker cannot obtain any information on the secret key K, irrespective on the amount of ciphertexts he's being checking on. I.e., the cipher's security is not diminishing at increase of the total size of the encrypted plaintext prior to the secret key change. It follows from the feature "random input - random output" that each non-expanding cipher becomes perfectly secure if BSS is the source of plaintexts. This means, for such sources MV2 is a perfectly secure cipher.

7.4 Testing of the algorithm

Testing on correspondence to dependence criteria

For testing a basic implementation of the MV2 algorithm on correspondence to dependence criteria the formulae (5.10), (5.11) and (5.12) were used, and also (5.13) and (5.14).

The crytpographic primitive MV2 doesn't belong to neither block class ciphers. It is an iterative probabilistic cipher, where each iteration resembles a round of a substitution-permutation network when not a single block as in block ciphers, but rather the whole message is being processed.

The lengths of outputs of MV2 are dependent on a plintext M, a key K and a randomizer R. In this case the normalizing coefficient is not completely correct into expressions (5.11), (5.12), (5.13) and (5.14). This normalization understates values of the degree of avalanche effect and the degree of strict avalanche criterion. But this criteria can be used for the testing of MV2.

The method of testing on corresponding to dependence criteria

We test MV2 with four variants of size of an input block. We consider the plaintext sizes of 16, 32, 64 and 128 bytes. For each variant, we consider 5000 randomly chosen inputs encrypted under a single randomly chosen MV2 key. These examinations is carried out for varying numbers of rounds, from 1 round to the 16-th rounds. Initial data were taken from a file containing a sequence generated by a physical random number generator.

We've determined the maximum length of outputs, average length of the core (L_c^*) and string of flags (L_f^*), the average number of output bits changed when changing 1 input bit, and separately for the core and the string of flags, the degree of completeness, the degree of avalanche effect (d_a^c and d_a^f) and the degree of the strict avalanche criterion (d_{sa}^c and d_{sa}^f) were computed.

The degree of completeness, except for one special modification of MV2 has always been ≈ 1.0

The algorithm's output is the core and the string of flags. The lengths of the core and the string of flags are changed on every round. The following expressions were used as the degree of avalanche effect and the degree of the strict avalanche criterion:

$$d_a = \frac{d_a^c \cdot L_c^* + d_a^f \cdot L_f^*}{L_c^* + L_f^*},$$

$$d_{sa} = \frac{d_{sa}^c \cdot L_c^* + d_{sa}^f \cdot L_f^*}{L_c^* + L_f^*}.$$

Dependence of d_a and d_{sa} on size of input data

A substitution transformation (see section 3.2) for each entry byte assigns a reminder, which has the length from 3 to 7 bits. If a plaintext is short, only some part of a substitution table is used for the text transformation on every round; besides, the 128-bit fixed permutation is used, which doesn't work for texts less than 16 bytes long, therefore,we may assume that values d_a and d_{sa} must depend on the size of entry data.

Fig. 7.1 and 7.2 show values of d_a and d_{sa} at different input lengths. It's obvious that with the growth of input lengths, the values of d_a and d_{sa} are growing as well for all rounds.

On the choice of a permutation transformation

There are some mechanisms for providing diffusion in the round function. In the basic realization of the MV2 algorithm a 128-bit linear transformer is used to ensure dispersion. Further we'll call it as basic one.

To determine the influence of a permutation transformation, we made changes to the source code of the algorithm and computed values d_a and d_{sa} for different permutation transformations. We've considered the following variants:

1) with a basic permutation transformation;

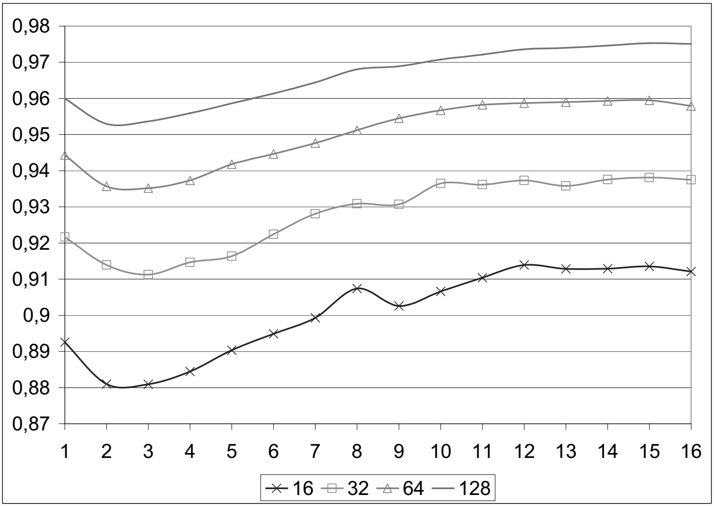

Figure 7.1: *Comparison of criteria d_a for 128, 64, 32 and 16 -byte inputs*

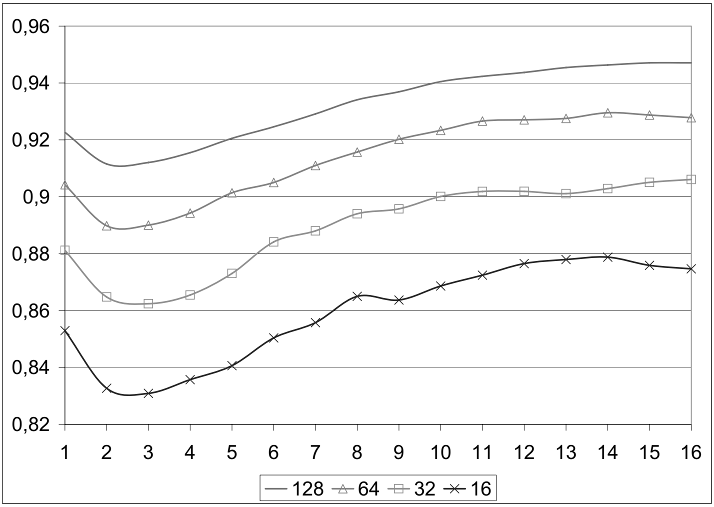

Figure 7.2: *Comparison of criteria d_{sa}(strict avalanche effect) for 128, 64, 32 and 16 -byte inputs*

2) with the affine byte permutation transformation;

3) with "armenian" shuffle [31].

4) without a permutation transformation;

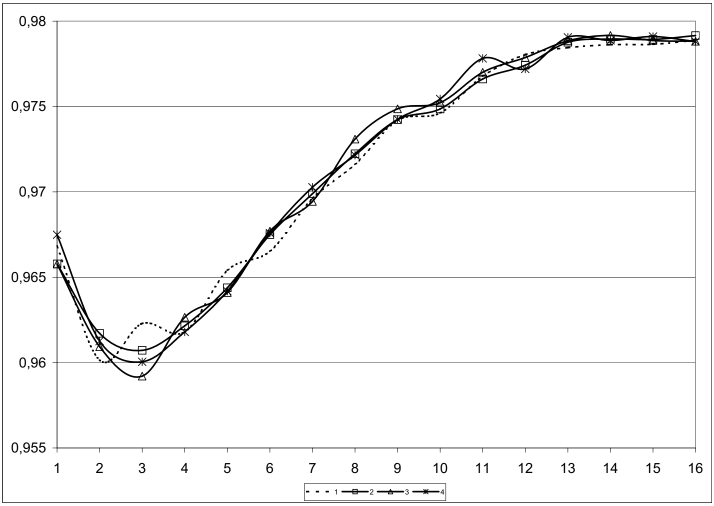

Figure 7.3: *Comparison of criteria of avalanche effect d_a for 128-bit random inputs at different permutations . 1 – "armenian", 2 – basic, 3 – affine permutation, 4 – without a permutation transformation;*

As the charts in Fig. 7.3 and 7.4 show, different permutations practically have no influence on the values d_a and d_{sa}.

We conclude that this effect is a corollary of the pseudorandom change of substitution transformations at each round. To check the hypothesis, the random number generator has been turned off in the algorithm's source code and a fixed sequence of numbers has been used for selection of the round substitution transformation. Thus, during all tests the same substitution transformations were made for the same rounds. For these conditions, degrees of avalanche, completeness and strict avalanche for 128-byte inputs were computed..

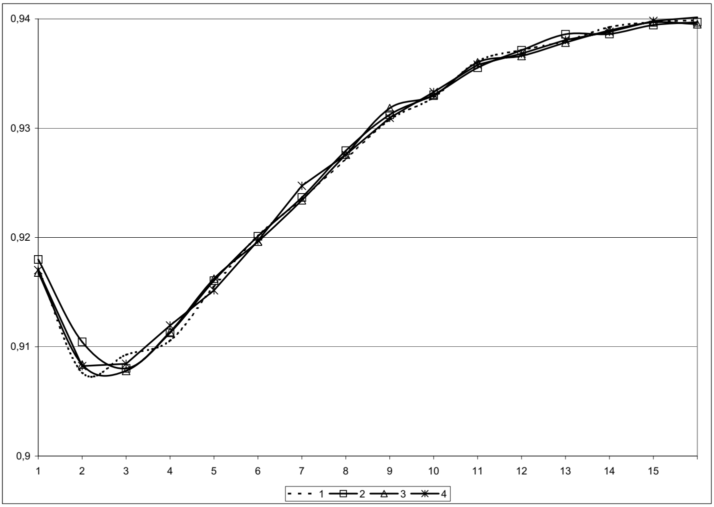

Figure 7.4: *Comparison of criteria of s strict avalanche effect d_{sa} for 128-bit random inputs at different permutations. 1 – ”armenian”, 2 – basic, 3 – affine permutation, 4 – without a permutation transformation;*

At these conditions degrees of completeness, avalanche and strict avalanche were computed for 128-bit inputs. The tests results are displayed in the charts in the Fig. 7.5 and 7.6. Upper charts (MV2) in Fig. 7.5, 7.6 and 7.7, 7.8 correspond to a usual (pseudorandom) table shuffle without a linear transformation before the substitution.

We can see from these charts, that a pseudorandom choice of substitution transformation at each round has greater influence on the values d_a and d_{sa}, than a permutation transformation.

As the difference between the values d_a and d_{sa} till the 4th round and after it is not very big (Fig. 7.5, 7.6), in Fig. 7.7 and 7.8 the values d_a and d_{sa},were concluded beginning with the 4th round.

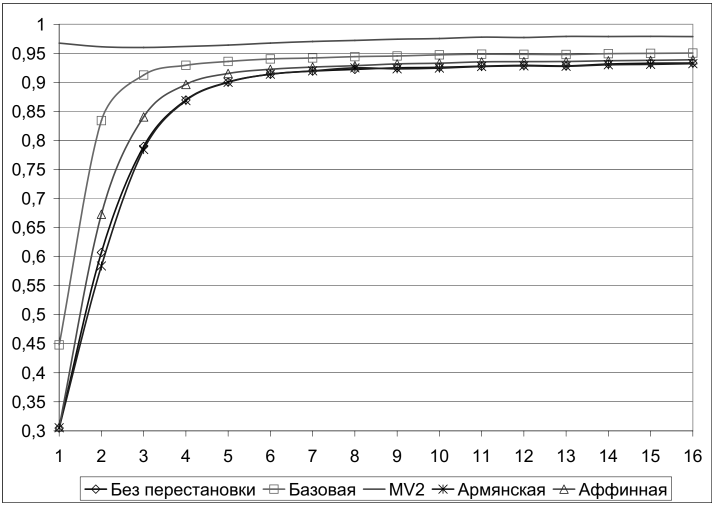

Figure 7.5: *Comparison of the criteria d_a for different permutations at the fixed substitution transformations*

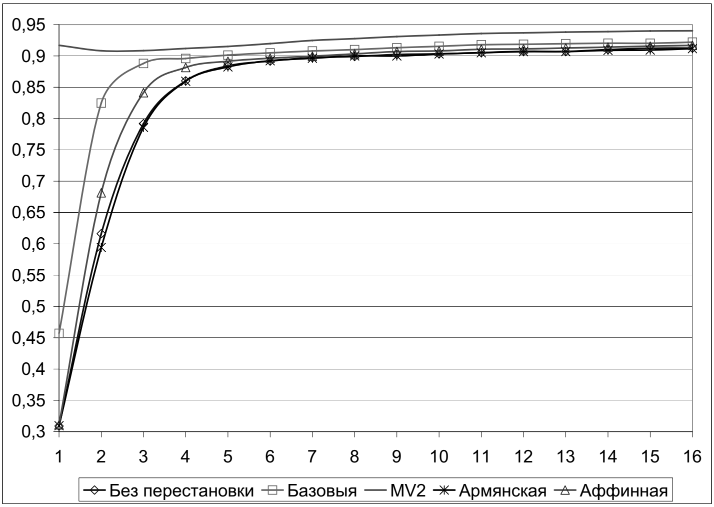

Figure 7.6: *Comparison of the criteria d_{sa} for different permutations at the fixed substitution transformations*

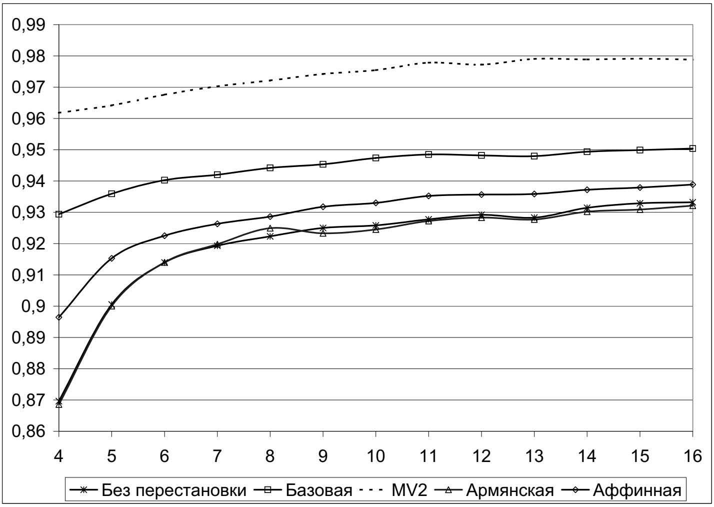

Figure 7.7: *Comparison of the criteria d_a for different permutation transformations at the fixed substitutions beginning with the 4th round*

One can see from the charts in Fig. 7.7 and 7.8, that the basic permutation has a better influence on the values d_a and d_{sa}.

The results of testing (Fig. 7.3 – 7.8) allow drawing conclusions about the following: firstly, a pseudorandom change of permutation transformation has bigger influence on values of dependence criteria, than that one of a linear transformation, and, secondly, a chosen in the basic realization permutation transformation is better than others considered ones.

Influence of whitening

The MV2 encryption algorithm is a cascade of a stream cipher and the general scheme of harming. Application of a stream cipher ensures whitening of a plaintext.

To check its impact on the degree of completeness, degrees d_a and d_{sa} we carry out tests for MV2 with whitening and MV2 without whitening (Fig. 7.9, 7.10). The first test con-

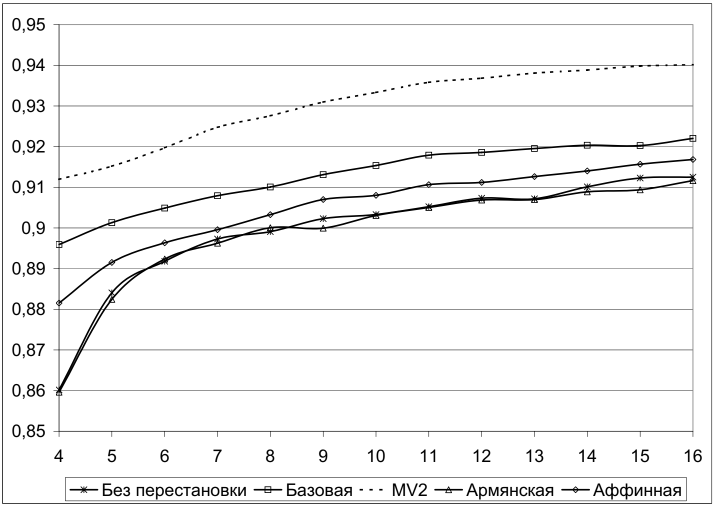

Figure 7.8: *Comparison of the criteria d_{sa} for different permutation transformations at the fixed substitutions beginning with the 4th round*

sist of 5000 randomly selected 16- and 128-byte bytes plaintexts. The second test consists of 256 128-byte homogeneous plaintexts $M = x^{128}$ where $x \in \{0,1\}^8$. Under a homogeneous input here we understand a sequence consisting of the same bytes. The following designations are used in these charts: 1a, 2a and 1b, 2b – correspondingly 16- (1), 128-byte (2) random inputs at presence (a) or absence (b) of whitening, 3a, 4a and 3b, 4b – correspondingly 16- (3), 128-byte (4) homogeneous inputs at presence (a) or absence (b) of whitening.

It follows from the charted displayed in Fig. 7.9 and Fig. 7.10, that whitening has a significant impact on values of degrees of avalanche criteria and a strict avalanche criteria in case of homogeneous inputs. This impact grows at increase of the input length.

In our attacks on MV2 we discovered that whitening substantially increased the difficulty of attacking the cipher, by hiding from a attacker the specific inputs to the first round.

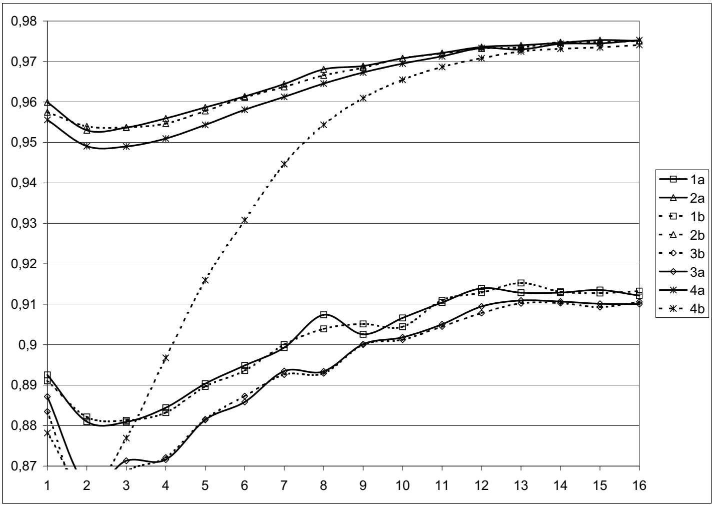

Figure 7.9: *Dependence $d_a$on presence of whitening and type of input data at 16- and 128-byte inputs:*

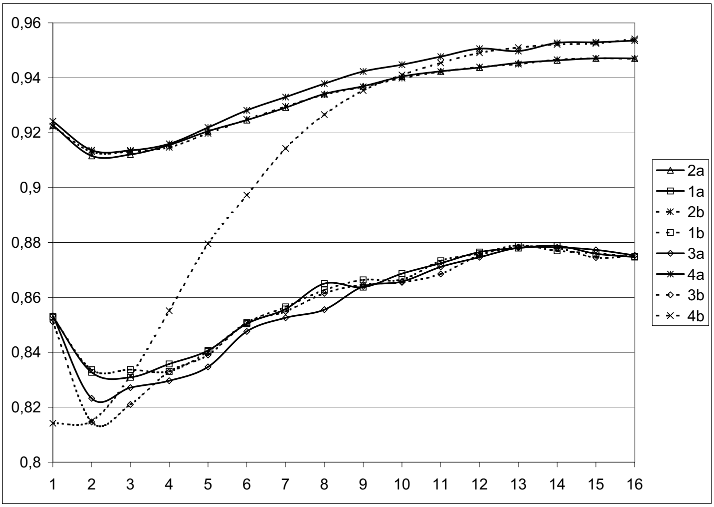

Figure 7.10: *Dependence d_{sa} on presence of whitening and type of input data at 16- and 128 byte inputs*

Whitening and the flag output

We also carried out testing to evaluate impact of whitening on the flag output. As the mappings satisfying SAC, satisfy other criteria as well, then the flag output was tested for correspondence on a strict avalanche criterion. The degree of a strict avalanche criteria was defined by the formula (5.14). At computing according to this formula it's possible to estimate an error occurred due to a variable input length.

The charts of degree of a SAC of the flag output on the number of rounds for different input lengths and tests are showed in Fig. 7.11.

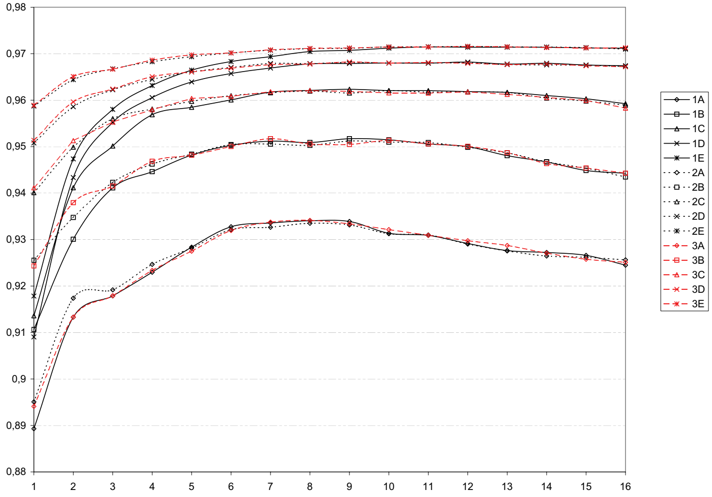

Figure 7.11: *Charts of dependence degree of a SAC of the flag output on the number of rounds for different output lengths: 1 – homogeneous inputs without whitening; 2 – homogeneous inputs with whitening; 3 – random inputs without whitening; A – 16, B – 32, C – 64, D – 128, E – 256-byte inputs*

Sequences consisting of 16, 32, 64, 128 and 256 bytes went to the input. For each input length three groups of tests of computing the degree of a strict avalanche criterion were carried out:

1. A test for homogeneous inputs without whitening (Fig. 7.11, charts 1A – 1E);

2. A test for homogeneous inputs with whitening (Fig. 7.11, charts 2A – 2E);

3. A test for random inputs without whitening (Fig. 7.11, charts 3A – 3E).

From the comparison of charted displayed in Fig. 7.11, we can draw conclusions about the following:

- values of the SAC degree grow at increasing an input length;

- at increasing a number of rounds values of a SAC degree start decreasing for short inputs (16, 32, 64 byte).

Usually, the MV2 algorithm is used to encrypt data of large capacity, therefore the most interesting case is to consider values of SAC degrees for 256-byte inputs. The corresponding charts are represented in Fig. 7.12.

We can see from the charts displayed in Fig. 7.12, that:

- values of a SAC degree at homogeneous inputs and whitening behave in the same way as at a random input without whitening;

- at first rounds if there’s no whitening, the values of a SAC degree are considerably smaller than in the case with whitening.

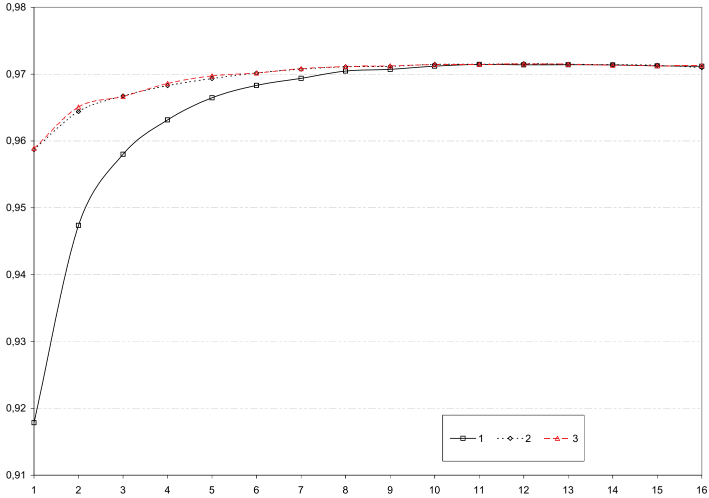

Figure 7.12: *Charts of dependence of a SAC degree of the flag output on a number of flags for 256-byte inputs: 1 – homogeneous inputs without whitening; 2 – homogeneous inputs with whitening; 3 – random inputs without whitening*

From the charts (Fig. 7.11 and 7.12) we can see that whitening has a considerable impact on values of degree of an avalanche and a strict avalanche of the flag output in case of homogeneous outputs. Whitening considerably increases the difficulty of attacking by known flags by concealing from an attacker the specific inputs to the first round.

For 256-byte inputs degrees of a SAC of various texts began to coincide after 7 transformation rounds. Consequently, we can draw a conclusion that for long input texts it's recommended to perform no less than 7 encryption rounds. For short input texts (less than 128 byte), on the contrary, it's not recommended to perform more than 10 encryption rounds. Consequently implementation of the algorithm should be built in such a way that wouldn't allow short (less that 16 bytes) remainder outputs, at that no less that 8 transformation rounds should be performed.

The length of the flag output

In the table. 7.2 there are average values of flags output lengths. The values at which the average output length is bigger than an input length are in bold type. The correlation of the table with the charts in Fig. 7.12 allows drawing the conclusion that the optimal number of rounds depends on a length of an input text.

Table 7.2: *Tentative and calculated (in the parenthesis) average lengths of the flag output (in bytes) at different input lengths for varying number of rounds*

	16	*32*	*64*	*128*	*256*
1	5,0(6,2)	8,9(13,0)	16,6(26,7)	32,1(53,9)	63,0(108,4)
2	8,9(9,8)	15,7(18,8)	29,3(36,9)	56,5 (73,0)	110,9(145,3)
3	12,3(12,9)	21,3(23,6)	39,4(45,1)	75,4 (88,0)	147,6(173,7)
4	15,3(15,7)	26,0(27,7)	47,4(51,7)	90,2 (99,7)	175,9(195,7)
5	**18,1**(18,2)	30,1(31,2)	54,0(57,1)	101,9(109,0)	197,7(212,8)
6	20,6(20,6)	**33,57**(34,3)	59,4(61,7)	111,2(116,5)	214,8(226,1)
7	23,0(22,8)	36,7(37,0)	**64,0**(65,6)	118,7(122,6)	228,1(236,7)
8	25,2(24,9)	39,5(39,6)	67,9(68,9)	124,9(127,6)	238,7(245,1)
9	27,4(26,9)	42,1(41,9)	71,4(71,9)	**130,0**(131,9)	247,2(251,9)
10	29,5(28,8)	44,5(44,1)	74,4(74,6)	134,3(135,5)	254,1(257,5)
11	31,6(30,7)	46,8(46,2)	77,18(77,0)	138,0(138,7)	**259,7**(262,1)
12	33,6(32,6)	48,9(48, 2)	79,7(79, 3)	141,3(141,6)	264,5(266,1)
13	35,6(34,5)	51,1(50,1)	82,1(81,5)	144,3(144,2)	268,5(269,5)
14	37,5(36,3)	53,1(52,0)	84,4(83,5)	146,9(146,5)	272,0(272,5)
15	39,5(38,1)	55,1(53,9)	86,5(85,5)	149,4(148,8)	275,2(275,2)
16	41,4(39,9)	57,1(55,7)	88,7(87,5)	151,7(150,9)	278,0(277,7)

The criterion of completeness

The dependencies tests show that MV2 satisfy to the completeness criterion.

During examinations tests in which at each round a certain substitution transformation was performed were carried out. In these tests we found the index of dependence of the degree of completeness on a number of performed rounds. In

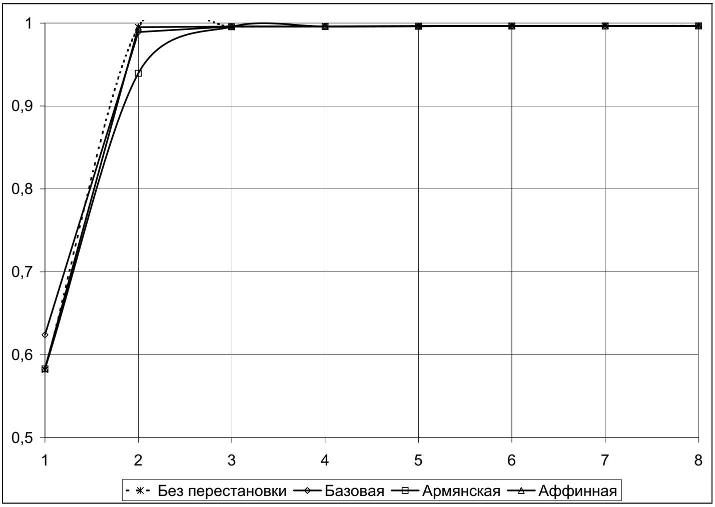

Figure 7.13: *Comparison of degree of completeness for different linear transformations at the fixed set of substitution transformations and 128-byte random inputs*

Fig. 7.13 there are charts of values of the index of the degree of completeness at the fixed at each round number of substitution transformation for various types of permutation transformation. We can see from this chart that at the fixed extract of substitution transformation the criterion of completeness is carried out beginning with the 3d round.

Analysis of cores' output

The core is a harmed ciphertext.

In the table. 7.3 and 7.4 there are experimental data obtained during testing the core output on accordance to dependence criteria. In these tables the following designations are used: *Rnd* is a number of a round, $\overline{L}_C$ is the maximal length of the core output (in bits), $\widetilde{L}_C$ is the average length of the core output (in bits), Δ – the average number of changed bits, d_c is degrees of completeness, d_a and $\overline{d}_{sa}$ – correspondingly degrees of avalanche and strict avalanche, computed by (5.11) and (5.12), $\widetilde{d}_{sa}$ is the degree of a strict avalanche, calculated by (5.12), σ is an experimental value of a standard deviation of the core length from the average value and σ/m – evaluation of the error of the degree of a strict avalanche calculated by (5.14).

Evaluations of errors

Normalization factor in the expressions (5.11), (5.13) and (5.12),(5.14) is not used entirely correctly. Let's evaluate an error occurred due to incorrect evaluation of "tails" in (5.14). Denote μ the average output length. Then from (5.14) we have

$$\widetilde{d}_{sa} = 1 - \frac{1}{mn}\sum_{i=1}^{n}\sum_{j=1}^{\mu}\left|\frac{2a_{ij}}{\#\mathfrak{X}} - 1\right| - \frac{1}{mn}\sum_{i=1}^{n}\sum_{j=\mu+1}^{\widetilde{m}_i}\left|\frac{2a_{ij}}{\#\mathfrak{X}} - 1\right| \tag{7.7}$$

For the component

$$\overline{d_{sa}} = 1 - \frac{1}{mn}\sum_{i=1}^{n}\sum_{j=1}^{\mu}\left|\frac{2a_{ij}}{|X|} - 1\right|$$

the value a_{ij} can be considered correct, and the second one in (7.7) is computed with an error.

Then the real value

$$d^{*}_{sa} = \overline{d_{sa}} + \delta,$$

where δ – is an added error.

Let's consider a random value τ, which equals the output length. Let μ be an expectation and σ will be a variance of

Table 7.3: *Results of testing dependence criteria for 16-byte inputs*

Rnd	$\overline{L}_C$	$\widetilde{L}_C$	Δ	d_c	d_a	$\overline{d}_{sa}$	$\widetilde{d}_{sa}$	σ	σ/m
1	127	109,9	60,83	1	0,96	0,88	0,95	7,68	0,07
2	119	93,65	53,48	1	0,94	0,86	0,94	8,08	0,09
3	111	81,30	47,55	1	0,92	0,83	0,94	8,11	0,1
4	103	71,94	43,03	1	0,91	0,81	0,93	7,93	0,11
5	95	65	39,20	1	0,9	0,8	0,93	7,74	0,12
6	90	59,61	36,67	1	0,9	0,8	0,92	7,52	0,13
7	85	55,59	34,48	1	0,88	0,78	0,92	7,37	0,13
8	79	52,49	32,72	1	0,89	0,78	0,92	7,16	0,14
9	79	50,27	31,57	1	0,89	0,78	0,92	7,0	0,14
10	75	48,41	30,33	1	0,88	0,77	0,92	6,81	0,14
11	73	47,13	29,73	1	0,89	0,78	0,91	6,7	0,14
12	71	45,92	28,99	1	0,9	0,8	0,92	6,56	0,14
13	71	45,17	28,45	1	0,9	0,79	0,91	6,45	0,14
14	70	44,61	28,15	1	0,89	0,78	0,91	6,39	0,14
15	64	44,16	27,77	1	0,89	0,76	0,91	6,3	0,14
16	63	43,75	27,64	1	0,89	0,75	0,91	6,29	0,14

Table 7.4: *Results of testing dependence criteria for 64-byte inputs*

Rnd	$\overline{L}_C$	$\widetilde{L}_C$	Δ	d_c	d_a	$\overline{d}_{sa}$	$\widetilde{d}_{sa}$	σ	σ/m
1	447	400,3	209,2	1	0,97	0,93	0,97	20,6	0,05
2	366	313,8	167,2	1	0,95	0,9	0,96	18,4	0,06
3	303	248,0	134,7	1	0,94	0,89	0,96	16,3	0,07
4	249	198,5	109,7	1	0,93	0,87	0,95	14,7	0,07
5	207	160,8	90,33	1	0,92	0,85	0,95	13,3	0,08
6	178	132,1	75,58	1	0,91	0,84	0,94	11,9	0,09
7	157	110,7	64,2	1	0,9	0,83	0,94	10,9	0,1
8	134	94,06	55,62	1	0,9	0,82	0,93	10,0	0,11
9	119	81,73	48,7	1	0,89	0,81	0,93	9,24	0,11
10	109	72,32	43,47	1	0,89	0,8	0,93	8,64	0,12
11	103	65,06	39,71	1	0,89	0,8	0,92	8,16	0,13
12	94	59,73	36,74	1	0,89	0,8	0,92	7,79	0,13
13	87	55,7	34,57	1	0,88	0,79	0,92	7,5	0,13
14	85	52,46	32,76	1	0,88	0,8	0,92	7,26	0,14
15	79	50,33	31,56	1	0,88	0,78	0,92	7,03	0,14
16	79	48,57	30,44	1	0,87	0,78	0,92	6,86	0,14

this random value. As $a_{ij} = 0, j > \mu$, then we can assume that

$$\delta \approx \frac{1}{mn} \sum_{i=1}^{n} \sum_{j=\mu+1}^{\overline{m}_i} 1.$$

Deviations from the average length by the value which is bigger than variance are unlikely, then $\overline{m}_i - \mu \approx \sigma$ and $\delta \approx \sigma/m$.

From the given tables 7.3 and 7.4 it follows that the value of degree of a strict avalanche is close to 1. Thus, the MV2 algorithm satisfies dependence criteria.

Statistical testing

Using the battery of Diehard tests, flags of long files have been tested. The tests results allow for interpretation of flags as "random" sequences.

Core length

Deviation of expectation value of the core length from the real one is less than 1 byte for short plaintext and 1% for the long ones.

Flag length

Deviation of expectation value of the flags length from the real one is less than 1 byte for short plaintext and 1% for the long ones.

Performance

During testing we made 100 encryptions of 200000 byte pseudorandom sequence with different keys. The text module has been compiled by MS VC 7.0 compiler. Testing was made on a PC with a Pentium 4 processor, 1700MHz, RAM 256 MB, 266 MHz DDR, Windows XP. Encryption rate achieved was ≈ 5 MB/sec.

A compiler significantly impacts the algorithm's implementation speed.

Appendix A The pseudocode of the key schedule of the MV2 cipher

A.1 Key schedule of the cipher MV2-128

The key schedule of MV2 processes the initial 128-bit master key into 32 256-byte subkeys. A subkey is a permutation with values from 0 to 255.

The basic data unit in the scheduling is a 32 bit word. The key scheduling algorithm extends 4-word (128-bit) key $key[0], key[1], key[2], key[3]$ into an array of words $kx[0]$, $kx[1], \ldots, kx[255]$. Finally these words are translated into 32 key substitutions $p[0,\cdot], p[1,\cdot], \ldots p[31,\cdot]$ required by MV2.

The key schedule algorithm can be described by the pseudo-code:

```
static long t[8]=
{       0x726a8f3b, 0xe69a3b5c, 0xd3c71fe5, 0xab3c73d2,
        0x4d3a8eb3, 0x0396d6e8, 0x3d4c2f7a, 0x9ee27cf3 };
```

```
$for(i = 0; i < 4; i++)kx[i] = key[i];$
$for(i = 4; i < 256; i++)$ {
        $s0 = g(kx[i-1], kx[i-2], t[0]);$
        $s1 = g(kx[i-1] \lll 3, kx[i-3], t[1]);$
        $s2 = g(kx[i-1] \lll 7, kx[i-4], t[2]);$
        $s3 = g(kx[i-2] \lll 3, kx[i-3] \lll 3, t[3]);$
        $s4 = g(kx[i-2] \lll 7, kx[i-4] \lll 3, t[4]);$
        $s5 = g(kx[i-3] \lll 7, kx[i-4] \lll 7, t[5]);$
        $s6 = g(kx[i-1], kx[i-2] \lll 11, kx[i-3] \lll 17) + t[6] \lll (s0 + s1 + s2)\&15;$
        $s7 = g(kx[i-1] \lll 2, kx[i-2] \lll 17, kx[i-4]) + t[7] \lll (s3 + s4 + s5)\&15;$
        $s8 = g(s2, s3, s4) \lll 2;$
        $s9 = g(s0, s5, s6) \lll 11;$
        $s10 = g(s1, s2 \lll 14, s7) \lll 13;$
        $s11 = g(s3 \lll 6, s4 \lll 4, s5 \lll 12) \lll 9;$
        $s12 = g(s0 \lll 7, s1 \lll 17, s6 \ggg 12) \lll 3;$
        $s13 = g(s2 \lll 10, s4 \lll 12, s7 \lll 16) \lll 7;$
        $s14 = g(s0 \lll 13, s3 \lll 7, s5 \lll 11) \lll 16;$
        $s15 = g(s1 \lll 5, s6 \lll 12, s7 \lll 10) \lll 5;$
        $s3 = s8 \oplus s9 \oplus s10 \oplus s11 \oplus s12 \oplus s13 \oplus s14 \oplus s15;$
        $s3 = s3 + s3 \lll 11;$
        $s3 = s3 \oplus s3 \lll 5;$
        $kx[i] = s3;$
}
for (i=0;i¡256;i++) {
    $kx[i] = kx[i] \oplus ((kx[(i+23)\&255]) + kx[kx[(i+19)\&255]\&255]);$
    for (j=0;j¡32;j++)
    $p[j, i] = i;$
}
for (i=255;i¿0;i– –) {
     $x = kx[i] + (kx[(i+113)\&255] \lll 11);$
    for(j=0;j¡32;j++) {
        $k = (x\&255)\%(i+1);$
        $tmp = p[j, k];$
        $p[j, k] = p[j, i];$
        $p[j, i] = tmp;$
        $s0 = g(s0, s2, kx[i + j + 47] \lll ((j + 3)\&15));$
        $s7 = g(s7, s1, kx[i + j + 59] \lll ((j + 3)\&15));$
        $s4 = g(s4, s3, kx[i + j + 67] \lll ((j + 3)\&15));$
        $s6 = g(s6, s5, kx[i + j + 73] \lll ((j + 3)\&15));$
        $s2 = g(s2, s1, kx[i + j + 83] \lll ((j + 3)\&15));$
        $s3 = g(s3, s5, kx[i + j + 97] \lll ((j + 3)\&15));$
        $s5 = g(s5, s1, kx[i + j + 103] \lll ((j + 3)\&15));$
        $s1 = g(s1, s0, kx[i + j + 109] \lll ((j + 3)\&15));$
        $s3 = s3 - s0;$
        $s6 = s6 + s2;$
        $s7 = s7 \oplus s5;$
        $s1 = s1 - s4;$
        $s0 = s0 - s7;$
        $s4 = s4 \oplus s6;$
        $s2 = s2 + s3;$
        $s5 = s3 + s1;$
        $s1 = s1 + (s7 \lll 11);$
        $s3 = s3 \oplus (s6 \lll 17);$
        $s2 = s2 - (s5 \lll 13);$
        $s0 = s0 + (s4 \lll (s3\&15));$
        $s1 = s1 \oplus s3;$
        $s0 = s0 \oplus (s2 \lll 5);$
        $x = x * (x \ggg 1) + (x \ggg 17) \oplus (kx[(i + j + 127)\&255] \lll (j\&15)) \oplus (s1 + s0);$
    }
}
```

A.2 Key schedule of the cipher MV2-256

The key schedule of MV2 processes the initial 256-bit master key into 32 256-byte subkeys. A subkey is a permutation with values from 0 to 255.

The basic data unit in the scheduling is a 32 bit word. The key scheduling algorithm extends 8-word (256-bit) key $key[0], key[1], \ldots, key[7]$ into an array of words $kx[0]$, $kx[1], \ldots,$ $kx[255]$. Finally these words are translated into 32 key substitutions $p[0,\cdot], p[1,\cdot], \ldots p[31,\cdot]$ required by MV2.

The key schedule algorithm can be described by the pseudocode:

```
static long t[8]=
{
        0x726a8f3b, 0xe69a3b5c, 0xd3c71fe5, 0xab3c73d2,
        0x4d3a8eb3, 0x0396d6e8, 0x3d4c2f7a, 0x9ee27cf3
};
for (i=0; i¡8; i++) kx[i]=key[i];
for (i=8;i¡256;i++) {
        $s0 = (((kx[i-4] \oplus kx[i-3]) \& kx[i-2]) \oplus kx[i-3]) + t[0] + kx[i-1] \lll 13;$
        $s1 = (((kx[i-8] \oplus kx[i-7]) \& kx[i-6]) \oplus kx[i-7]) + t[1] + kx[i-5] \lll 13;$
        $s2 = (((kx[i-6] \oplus kx[i-5]) \& kx[i-4]) \oplus kx[i-5]) + t[2] + kx[i-3] \lll 13;$
        $s3 = (((kx[i-1] \oplus kx[i-2]) \& kx[i-8]) \oplus kx[i-2]) + t[3] + kx[i-7] \lll 13;$
        $s4 = (((kx[i-8] \oplus kx[i-6]) \& kx[i-4]) \oplus kx[i-6]) + t[4] + kx[i-2] \lll 13;$
        $s5 = (((kx[i-7] \oplus kx[i-5]) \& kx[i-3]) \oplus kx[i-5]) + t[5] + kx[i-1] \lll 7;$
        $s6 = t[6] + kx[i-7]kx[i-4] \oplus kx[i-6]kx[i-2] \oplus kx[i-5]kx[i-3] \oplus kx[i-2]kx[i-1] \oplus kx[i-1];$
        $s7 = t[7] + kx[i-8]kx[i-6]kx[i-4]kx[i-2] \oplus kx[i-8]kx[i-7] \oplus kx[i-6]kx[i-5] \oplus kx[i-5]kx[i-4] \oplus kx[i-3]kx[i-2] \oplus kx[i-4];$
        $s8 = g(s2, s3, s4) \lll 2;$
        $s9 = g(s0, s5, s6) \lll 11;$
        $s10 = g(s1, s2 \lll 14, s7) \lll 13;$
        $s11 = g(s3 \lll 6, s4 \lll 4, s5 \lll 12) \lll 9;$
        $s12 = g(s0 \lll 7, s1 \lll 17, s6 \ggg 12) \lll 3;$
        $s13 = g(s2 \lll 10, s4 \lll 12, s7 \lll 16) \lll 7;$
        $s14 = g(s0 \lll 13, s3 \lll 7, s5 \lll 11) \lll 16;$
        $s15 = g(s1 \lll 5, s6 \lll 12, s7 \lll 10) \lll 5;$
        $s3 = s8 \oplus s9 \oplus s10 \oplus s11 \oplus s12 \oplus s13 \oplus s14 \oplus s15;$
        $s3 = s3 + s3 \lll 11;$
        $s3 = s3 \oplus s3 \lll 5;$
        $kx[i] = s3;$
}
for (i=0;i¡256;i++) {
    $kx[i] = kx[i] \oplus ((kx[(i+23)\&255]) + kx[kx[(i+19)\&255]\&255]);$
    for (j=0;j¡32;j++)
    $p[j, i] = i;$
}
for (i=255;i¿0;i– –) {
     $x = kx[i] + (kx[(i + 113)\&255] \lll 11);$
    for(j=0;j¡32;j++) {
        $k = (x\&255)\%(i + 1);$
        $tmp = p[j, k];$
        $p[j, k] = p[j, i];$
        $p[j, i] = tmp;$
        $s0 = g(s0, s2, kx[i + j + 47] \lll ((j + 3)\&15));$
        $s7 = g(s7, s1, kx[i + j + 59] \lll ((j + 3)\&15));$
        $s4 = g(s4, s3, kx[i + j + 67] \lll ((j + 3)\&15));$
        $s6 = g(s6, s5, kx[i + j + 73] \lll ((j + 3)\&15));$
        $s2 = g(s2, s1, kx[i + j + 83] \lll ((j + 3)\&15));$
        $s3 = g(s3, s5, kx[i + j + 97] \lll ((j + 3)\&15));$
        $s5 = g(s5, s1, kx[i + j + 103] \lll ((j + 3)\&15));$
        $s1 = g(s1, s0, kx[i + j + 109] \lll ((j + 3)\&15));$
        $s3 = s3 - s0;$
        $s6 = s6 + s2;$
        $s7 = s7 \oplus s5;$
        $s1 = s1 - s4;$
        $s0 = s0 - s7;$
        $s4 = s4 \oplus s6;$
        $s2 = s2 + s3;$
        $s5 = s3 + s1;$
        $s1 = s1 + (s7 \lll 11);$
        $s3 = s3 \oplus (s6 \lll 17);$
        $s2 = s2 - (s5 \lll 13);$
        $s0 = s0 + (s4 \lll (s3 \& 15));$
        $s1 = s1 \oplus s3;$
        $s0 = s0 \oplus (s2 \lll 5);$
        $x = x * (x \gg 1) + (x \gg 17) \oplus (kx[(i + j + 127) \& 255] \lll (j \& 15)) \oplus (s1 + s0);$
    }
}
```

A.3 Key schedule of the cipher MV2-512

The key schedule of MV2 processes the initial 512-bit master key into 32 256-byte subkeys. A subkey is a permutation with values from 0 to 255.

The basic data unit in the scheduling is a 32 bit word. The key scheduling algorithm extends 16-word (512-bit) key $key[0], key[1], \ldots, key[15]$ into an array of words $kx[0]$, $kx[1]$, ..., $kx[255]$. Finally these words are translated into 32 key substitutions $p[0,\cdot], p[1,\cdot], \ldots p[31,\cdot]$ required by MV2.

The key schedule algorithm can be described by the pseudocode:

```
static long t[8]=
{
        0x726a8f3b, 0xe69a3b5c, 0xd3c71fe5, 0xab3c73d2,
        0x4d3a8eb3, 0x0396d6e8, 0x3d4c2f7a, 0x9ee27cf3
};
for (i=0;i¡16;i++) kx[i]=key[i];
for (i=16;i¡256;i++) {
        $s0 = (((kx[i-4] \oplus kx[i-3])\&kx[i-2]) \oplus kx[i-3]) + t[0] + kx[i-1] \lll 3;$
        $s1 = (((kx[i-8] \oplus kx[i-7])\&kx[i-6]) \oplus kx[i-7]) + t[1] + kx[i-5] \lll 3;$
        $s2 = (((kx[i-12] \oplus kx[i-11])\&kx[i-10]) \oplus kx[i-11]) + t[2] + kx[i-9] \lll 3;$
        $s3 = (((kx[i-16] \oplus kx[i-15])\&kx[i-14]) \oplus kx[i-15]) + t[3] + kx[i-13] \lll 3;$
        $s4 = (((kx[i-6] \oplus kx[i-5])\&kx[i-4]) \oplus kx[i-5]) + t[4] + kx[i-3] \lll 13;$
        $s5 = (((kx[i-10] \oplus kx[i-9])\&kx[i-8]) \oplus kx[i-9]) + t[5] + kx[i-8] \lll 13;$
        $s6 = (((kx[i-14] \oplus kx[i-13])\&kx[i-12]) \oplus kx[i-13]) + t[6] + kx[i-12] \lll 13;$
        $s7 = (((kx[i-16] \oplus kx[i-15])\&kx[i-2]) \oplus kx[i-15]) + t[7] + kx[i-1] \lll 13;$
        $s8 = g(s2, s3, s4) \lll 2;$
        $s9 = g(s0, s5, s6) \lll 11;$
        $s10 = g(s1, s2 \lll 14, s7) \lll 13;$
        $s11 = g(s3 \lll 6, s4 \lll 4, s5 \lll 12) \lll 9;$
        $s12 = g(s0 \lll 7, s1 \lll 17, s6 \lll 20) \lll 3;$
        $s13 = g(s2 \lll 10, s4 \lll 12, s7 \lll 16) \lll 7;$
        $s14 = g(s0 \lll 13, s3 \lll 7, s5 \lll 11) \lll 16;$
        $s15 = g(s1 \lll 5, s6 \lll 12, s7 \lll 10) \lll 5;$
        $s3 = s8 \oplus s9 \oplus s10 \oplus s11 \oplus s12 \oplus s13 \oplus s14 \oplus s15;$
        $s3 = s3 + s3 \lll 11;$
        $s3 = s3 \oplus s3 \lll 5;$
        $kx[i] = s3;$
```

```
}
for (i=0;i¡256;i++) {
    $kx[i] = kx[i] \oplus ((kx[(i+23)\&255]) + kx[kx[(i+19)\&255]\&255]);$
    for (j=0;j¡32;j++)
    $p[j, i] = i;$
}
for (i=255;i¿0;i) {
     $x = kx[i] + (kx[(i + 113)\&255] \lll 11);$
    for (j=0;j¡32;j++) {
        $k = (x\&255)\%(i + 1);$
        $tmp = p[j, k];$
        $p[j, k] = p[j, i];$
        $p[j, i] = tmp;$
        $s0 = g(s0, s2, kx[(i + j + 47)\&255] \lll ((j + 3)\&15));$
        $s7 = g(s7, s1, kx[(i + j + 59)\&255] \lll ((j + 3)\&15));$
        $s4 = g(s4, s3, kx[(i + j + 67)\&255] \lll ((j + 3)\&15));$
        $s6 = g(s6, s5, kx[(i + j + 73)\&255] \lll ((j + 3)\&15));$
        $s2 = g(s2, s1, kx[(i + j + 83)\&255] \lll ((j + 3)\&15));$
        $s3 = g(s3, s5, kx[(i + j + 97)\&255] \lll ((j + 3)\&15));$
        $s5 = g(s5, s1, kx[(i + j + 103)\&255] \lll ((j + 3)\&15));$
        $s1 = g(s1, s0, kx[(i + j + 109)\&255] \lll ((j + 3)\&15));$
        $s3 = s3 - s0;$
        $s6 = s6 + s2;$
        $s7 = s7 \oplus s5;$
        $s1 = s1 - s4;$
        $s0 = s0 - s7;$
        $s4 = s4 \oplus s6;$
        $s2 = s2 + s3;$
        $s5 = s3 + s1;$
        $s1 = s1 + (s7 \lll 11);$
        $s3 = s3 \oplus (s6 \lll 17);$
        $s2 = s2 - (s5 \lll 13);$
        $s0 = s0 + (s4 \lll (s3\&15));$
        $s1 = s1 \oplus s3;$
        $s0 = s0 \oplus (s2 \lll 5);$
        $x = x * (x \gg 1) + (x \gg 17) \oplus (kx[(i + j + 127)\&255] \lll (j\&15)) \oplus (s1 + s0);$
    }
}
```

A.4 Key schedule of the cipher MV2-1024

The key schedule of MV2 processes the initial 1024-bit master key into 32 256-byte subkeys. A subkey is a permutation with values from 0 to 255.

The basic data unit in the scheduling is a 32 bit word. The key scheduling algorithm extends 32-word (1024-bit) key $key[0], key[1], \ldots, key[31]$ into an array of words $kx[0]$, $kx[1]$, …, $kx[255]$. Finally these words are translated into 32 key substitutions $p[0, \cdot], p[1, \cdot], \ldots p[31, \cdot]$ required by MV2.

The key schedule algorithm can be described by the pseudocode:

```
static long t[8]=
{
        0x726a8f3b, 0xe69a3b5c, 0xd3c71fe5, 0xab3c73d2,
        0x4d3a8eb3, 0x0396d6e8, 0x3d4c2f7a, 0x9ee27cf3
};
for (i=0;i¡32;i++) kx[i]=key[i];
for (i=32;i¡256;i++) {
        $s0 = (((kx[i-4] \oplus kx[i-3])\&kx[i-2]) \oplus kx[i-3]) + t[0] + kx[i-1] \lll 3;$
        $s1 = (((kx[i-8] \oplus kx[i-7])\&kx[i-6]) \oplus kx[i-7]) + t[1] + kx[i-5] \lll 3;$
        $s2 = (((kx[i-12] \oplus kx[i-11])\&kx[i-10]) \oplus kx[i-11]) + t[2] + kx[i-9] \lll 3;$
        $s3 = (((kx[i-16] \oplus kx[i-15])\&kx[i-14]) \oplus kx[i-15]) + t[3] + kx[i-13] \lll 3;$
        $s4 = (((kx[i-6] \oplus kx[i-5]) \& kx[i-4]) \oplus kx[i-5]) + t[4] + kx[i-3] \lll 13;$
        $s5 = (((kx[i-10] \oplus kx[i-9]) \& kx[i-8]) \oplus kx[i-9]) + t[5] + kx[i-8] \lll 13;$
        $s6 = (((kx[i-14] \oplus kx[i-13]) \& kx[i-12]) \oplus kx[i-13]) + t[6] + kx[i-12] \lll 13;$
        $s7 = (((kx[i-16] \oplus kx[i-15]) \& kx[i-2]) \oplus kx[i-15]) + t[7] + kx[i-1] \lll 13;$
        $s14 = g(kx[i-14], kx[i-13], kx[i-12]);$
        $s8 = (g(kx[i-32], kx[i-31], kx[i-30]) + g(s2, s3, s4)) \lll 2;$
        $s9 = (g(kx[i-29], kx[i-28], kx[i-27]) + g(s0, s5, s6)) \lll 11;$
        $s10 = (g(kx[i-26], kx[i-25], kx[i-24]) + g(s1, s2 \lll 14, s7)) \lll 13;$
        $s11 = (g(kx[i-23], kx[i-22], kx[i-21]) + g(s3 \lll 6, s4 \lll 4, s5 \lll 12)) \lll 9;$
        $s12 = (g(kx[i-20], kx[i-19], kx[i-18]) + g(s0 \lll 7, s1 \lll 17, s6 \lll 20)) \lll 3;$
        $s13 = (g(kx[i-17], kx[i-32], kx[i-21]) + g(s2 \lll 10, s4 \lll 12, s7 \lll 16)) \lll 7;$
        $s14 = (g(kx[i-17], kx[i-16], kx[i-15]) + g(s0 \lll 13, s3 \lll 7, s5 \lll 11)) \lll 16;$
        $s15 = (g(kx[i-16], kx[i-31], kx[i-1]) + g(s1 \lll 5, s6 \lll 12, s7 \lll 10)) \lll 5;$
        $s3 = s8 \oplus s9 \oplus s10 \oplus s11 \oplus s12 \oplus s13 \oplus s14 \oplus s15;$
        $s3 = s3 + s3 \lll 11;$
        $s3 = s3 \oplus s3 \lll 5;$
        $kx[i] = s3;$
}
for (i=0;i¡256;i++) {
    $kx[i] = kx[i] \oplus ((kx[(i+23)\&255]) + kx[kx[(i+19)\&255]\&255]);$
    for (j=0;j¡32;j++)
    $p[j,i] = i;$
}
for (i=255;i¿0;i–) {
     $x = kx[i] + (kx[(i+113)\&255] \lll 11);$
    for (j=0;j¡32;j++) {
        $k = (x\&255)\%(i+1);$
        $tmp = p[j,k];$
        $p[j,k] = p[j,i];$
        $p[j,i] = tmp;$
        $s0 = g(s0, s2, kx[(i+j+47)\&255] \lll ((j+3)\&15));$
        $s7 = g(s7, s1, kx[(i+j+59)\&255] \lll ((j+3)\&15));$
        $s4 = g(s4, s3, kx[(i+j+67)\&255] \lll ((j+3)\&15));$
        $s6 = g(s6, s5, kx[(i+j+73)\&255] \lll ((j+3)\&15));$
        $s2 = g(s2, s1, kx[(i+j+83)\&255] \lll ((j+3)\&15));$
        $s3 = g(s3, s5, kx[(i+j+97)\&255] \lll ((j+3)\&15));$
        $s5 = g(s5, s1, kx[(i+j+103)\&255] \lll ((j+3)\&15));$
        $s1 = g(s1, s0, kx[(i+j+109)\&255] \lll ((j+3)\&15));$
        $s3 = s3 - s0;$
        $s6 = s6 + s2;$
        $s7 = s7 \oplus s5;$
        $s1 = s1 - s4;$
        $s0 = s0 - s7;$
        $s4 = s4 \oplus s6;$
        $s2 = s2 + s3;$
        $s5 = s3 + s1;$
        $s1 = s1 + (s7 \lll 11);$
        $s3 = s3 \oplus (s6 \lll 17);$
        $s2 = s2 - (s5 \lll 13);$
        $s0 = s0 + (s4 \lll (s3\&15));$
        $s1 = s1 \oplus s3;$
        $s0 = s0 \oplus (s2 \lll 5);$
        $x = x * (x \gg 1) + (x \gg 17) \oplus (kx[(i+j+127)\&255] \lll (j\&15)) \oplus (s1 + s0);$
    }
}
```

A.5 Key schedule of the cipher MV2-2048

The key schedule of MV2 processes the initial 2048-bit master key into 32 256-byte subkeys. A subkey is a permutation with values from 0 to 255.

The basic data unit in the scheduling is a 32 bit word. The key scheduling algorithm extends 64-word (2048-bit) key $key[0], key[1], \ldots, key[63]$ into an array of words $kx[0]$, $kx[1]$, $\ldots$, $kx[255]$. Finally these words are translated into 32 key substitutions

$p[0,\cdot], p[1,\cdot], \ldots p[31,\cdot]$ required by MV2.

The key schedule algorithm can be described by the pseudocode:

```
static long t[8]=
{
    0x726a8f3b, 0xe69a3b5c, 0xd3c71fe5, 0xab3c73d2,
    0x4d3a8eb3, 0x0396d6e8, 0x3d4c2f7a, 0x9ee27cf3
};
for (i=0;i¡64;i++) kx[i]=key[i];
for (i=64;i¡256;i++) {
    $s0 = (((kx[i-5] \oplus kx[i-4])\&kx[i-3]) \oplus kx[i-2]) + t[0] + kx[i-1] \lll 3;$
    $s1 = (((kx[i-10] \oplus kx[i-9])\&kx[i-8]) \oplus kx[i-7]) + t[1] + kx[i-6] \lll 3;$
    $s2 = (((kx[i-15] \oplus kx[i-14])\&kx[i-13]) \oplus kx[i-12]) + t[2] + kx[i-11] \lll 3;$
    $s3 = (((kx[i-20] \oplus kx[i-19])\&kx[i-18]) \oplus kx[i-17]) + t[3] + kx[i-16] \lll 3;$
    $s4 = (((kx[i-25] \oplus kx[i-24])\&kx[i-23]) \oplus kx[i-22]) + t[4] + kx[i-21] \lll 13;$
    $s5 = (((kx[i-30] \oplus kx[i-29])\&kx[i-28]) \oplus kx[i-27]) + t[5] + kx[i-26] \lll 13;$
    $s6 = (((kx[i-35] \oplus kx[i-34])\&kx[i-33]) \oplus kx[i-32]) +$
```

```
$t[6] + kx[i-31] \lll 13;$
        $s7 = (((kx[i-40] \oplus kx[i-39])\&kx[i-38]) \oplus kx[i-37]) + t[7] + kx[i-36] \lll 13;$
        $s8 = (g(kx[i-43], kx[i-42], kx[i-41]) + g(s2, s3, s4)) \lll 2;$
        $s9 = (g(kx[i-46], kx[i-45], kx[i-44]) + g(s0, s5, s6)) \lll 11;$
        $s10 = (g(kx[i-49], kx[i-48], kx[i-47]) + g(s1, s2 \lll 14, s7)) \lll 13;$
        $s11 = (g(kx[i-52], kx[i-51], kx[i-50]) + g(s3 \lll 6, s4 \lll 4, s5 \lll 12)) \lll 9;$
        $s12 = (g(kx[i-55], kx[i-54], kx[i-53]) + g(s0 \lll 7, s1 \lll 17, s6 \lll 20)) \lll 3;$
        $s13 = (g(kx[i-58], kx[i-57], kx[i-56]) + g(s2 \lll 10, s4 \lll 12, s7 \lll 16)) \lll 7;$
        $s14 = (g(kx[i-61], kx[i-60], kx[i-59]) + g(s0 \lll 13, s3 \lll 7, s5 \lll 11)) \lll 16;$
        $s15 = (g(kx[i-64], kx[i-63], kx[i-62]) + g(s1 \lll 5, s6 \lll 12, s7 \lll 10)) \lll 5;$
        $s3 = s8 \oplus s9 \oplus s10 \oplus s11 \oplus s12 \oplus s13 \oplus s14 \oplus s15;$
        $s3 = s3 + s3 \lll 11;$
        $s3 = s3 \oplus s3 \lll 5;$
        $kx[i] = s3;$
}
for (i=0;i¡256;i++) {
    $kx[i] = kx[i] \oplus ((kx[(i+23)\&255]) + kx[kx[(i+19)\&255]\&255]);$
    for (j=0;j¡32;j++)
    $p[j,i] = i;$
}
for (i=255;i¿0;i–) {
     $x = kx[i] + (kx[(i+113)\&255] \lll 11);$
    for (j=0;j¡32;j++) {
        $k = (x\&255)\%(i+1);$
        $tmp = p[j,k];$
        $p[j,k] = p[j,i];$
        $p[j,i] = tmp;$
        $s0 = g(s0, s2, kx[(i+j+47)\&255] \lll ((j+3)\&15));$
        $s7 = g(s7, s1, kx[(i + j + 59)\&255] \lll ((j + 3)\&15));$
        $s4 = g(s4, s3, kx[(i + j + 67)\&255] \lll ((j + 3)\&15));$
        $s6 = g(s6, s5, kx[(i + j + 73)\&255] \lll ((j + 3)\&15));$
        $s2 = g(s2, s1, kx[(i + j + 83)\&255] \lll ((j + 3)\&15));$
        $s3 = g(s3, s5, kx[(i + j + 97)\&255] \lll ((j + 3)\&15));$
        $s5 = g(s5, s1, kx[(i + j + 103)\&255] \lll ((j + 3)\&15));$
        $s1 = g(s1, s0, kx[(i + j + 109)\&255] \lll ((j + 3)\&15));$
        $s3 = s3 - s0;$
        $s6 = s6 + s2;$
        $s7 = s7 \oplus s5;$
        $s1 = s1 - s4;$
        $s0 = s0 - s7;$
        $s4 = s4 \oplus s6;$
        $s2 = s2 + s3;$
        $s5 = s3 + s1;$
        $s1 = s1 + (s7 \lll 11);$
        $s3 = s3 \oplus (s6 \lll 17);$
        $s2 = s2 - (s5 \lll 13);$
        $s0 = s0 + (s4 \lll (s3\&17);$
        $s1 = s1 \oplus s3;$
        $s0 = s0 \oplus (s2 \lll 5);$
        $x = x * (x \gg 1) + (x \gg 17) \oplus (kx[(i + j + 127)\&255] \lll (j\&15)) \oplus (s1 + s0);$
    }
}
```

A.6 Avalanche characteristics of key schedules

In this section we present the results of the statistical testing performed on our proposed key schedules.

For the results of this section, we generated four sets of 2000 keys for each length of keys and tested each key for avalanche properties. The set of 2000 keys was generated by using the physical random number generator. For each set of 2000 keys, and for each property, we report the mean, standard deviation, and minimum and maximum of the values obtained.

Table A.1: *Avalanche results for 2000 randomly generated keys*

Key size	**degree of avalanche effect**	**degree of strict avalanche criterion**
256	0,99783	0,99541
512	0,99569	0,99342
1024	0,99655	0,99442
2048	0,99925	0,99659

The values for our key schedules (Table A.1) are close to the theoretical values.

We have done dependencies tests for MV2 cipher functions mapping the key space to the ciphertext space under fixed 256-bytes inputs. For each variant of key size we select a fixed 256-byte input and encrypt it with 3200 randomly chosen keys. These examinations are carried out for various numbers of rounds, from 1 round to the 16-th rounds. Keys were taken from a file containing a sequence generated by a physical random number generator. The testing results for key lengths of 256, 512, 1024 and 2048 bits are shows on Fig.A.1(lines 1-4) . The line 5 on Fig. A.1 is supplied for compare. This line corresponds to the degree of SAC for the function from the plaintext space to the ciphertext space under a randomly fixed key and 3200 randomly selected 256-bytes plaintexts.

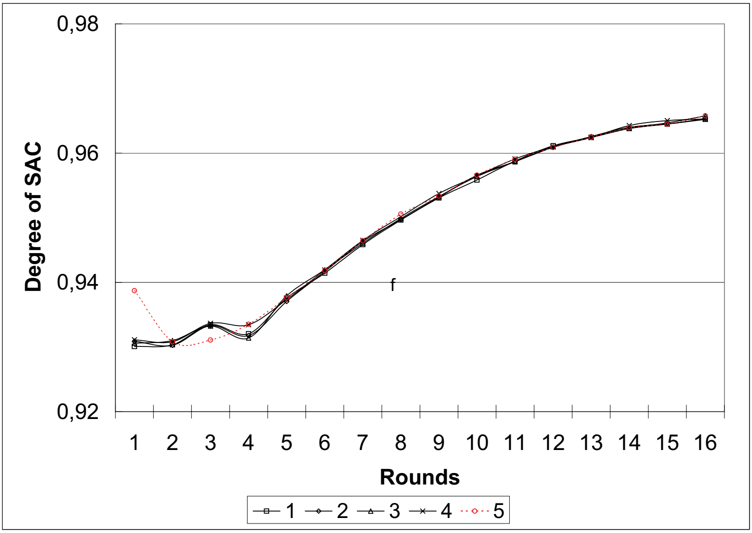

Figure A.1: *Degrees of SAC for different length of key.*

The test results for the standard avalanche properties of key schedules suggest that our key schedules have cryptographically good randomness properties. Not only do the subkeys generated perform well when tested for the standard avalanche properties, but there is evidence that the subkeys are not pairwise correlated.

Bibliography

[1] *Adams C.M. and Tavares S.E.* The Structured Design of Cryptographically Good SBoxes. // Journal of Cryptology V. 3, N. 1, P. 27 –41. 1990.

[2] *Adams C.M. and Tavares S.E.* Designing S-Boxes for Ciphers Resistant to Differential Cryptanalysis. // Proceedings of the 3rd Symposium on State and Progress of Research in Cryptography Rome, Italy, 15-16 Feh 1993, P. 181-190.

[3] *Andersen R., Biham E. and Knudsen L.* Serpent: A Proposal for Advanced Encryption Standard // First Advanced Encryption Standard (AES) Conference, Ventura, CA), 1998.

[4] *E. Biham and A. Shamir*, Differential cryptanalysis of DES-like cryptosystems, // Journal of Cryptology, Vol. 4, No. 1, pp. 3–72, 1991.

[5] *E. Biham and A. Shamir.* Differential Cryptanalysis of the Data Encryption Standard. Springer-Verlag, 1993.

[6] *Leon Brillouin.* Science and Information Theory. Academic Press, New York, 1956.

[7] *Chabaud F. and Vaudenay S.* Links between differential and linear cryptanalysis. *In A. De Santis, editor, Advances in Cryptology – Eurocrypt'94: Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 1994. Proceedings, volume 950 of Lecture Notes in Computer Science*, pages 356–365. Springer- Verlag, 1995.

[8] *Cover T.M. and King R.C.*, A Convergent Gambling Estimate of the Entropy of English // IEEE Transactions on Information Theory, V. IT-24, #. 4, Jul 1978, P. 329 – 421.

[9] J. Daemen and V. Rijmen. The Design of Rijndael. Information Security and Cryptography. Springer, 2002.

[10] . *Davies D.W. and Price W.L.* The Application of Digital Signatures Based on Public-Key Cryptosystems // Proceedings of the Fifth International Computer Communications Conference, Oct 1980, P. 525 – 530.

[11] *Dawson M.H. and Tavares S.E.* An Expanded Set of S-Box Design Criteria Based on Information Theory and Its Relation to Differential-like Attacks // Advances in Cryptology EUROCRYPT '91 Proceedings, Springer-Verlag, 1991, pp. 352-367.

[12] *Dawson M.H. and Tavares S.E.*, An Expanded Set of Design Criteria for Substitution Boxes and Their Use in Strengthening DES-Like Cryptosystems // IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing, Victoria, BC, Canada, 9-10 May 1991, P. 191-195.

[13] *Desmedt Yvo* Some Research Aspects of Threshold Cryptography // Information Security First International Workshop, ISW'97, Tatsunokuchi, Ishikawa Japan, September 17-19, 1997, Proceedings Series: Lecture Notes in Computer Science, V. 1396 Okamoto, Eiji; Davida, George; Mambo, Masahiro (Eds.) 1998, XII, P. 357, Softcover

[14] *Deavours C.A.* Unicity Points in Cryptanalysis // Cryptologia, V.1, N.1, P. 46 – 68.

[15] *Feistel H.* Cryptography and Computer Privacy. // Scientific American, May 1973, V. 228, N.5 P. 15 –23.

[16] *Feller W.* An Introduction to Probability Theory and Its Applications, Vol. 1, 3rd ed. New York: Wiley, 1968.

[17] *R. Gallager*, Information Theory and Reliable Communication. Notes from CISM, Udine, Italy: Springer-Verlag, 1972.

[18] *J.A. Gordon and R. Retkin*, "Are Big S-boxes Best?" // Cryptography, Proceedings of the Workshop on Cryptography, Burg Feuerstein, Germany, March 29-April 2, 1982, Springer-Verlag, 1983, pp. 257-262.

[19] *Hellman M.E.*, An Extension of the Shannon Theory Approach to Cryptography // IEEE Transactions on Information Theory, V.IT-23, N.3, May 1977, P. 289-294.

[20] *Heys H.M.*, The design of substitution-permutation network ciphers resistant to cryptanalysis, Ph.D. Thesis, Queen's University, Canada, 1994.

[21] *Heys H.M. and Tavares S.E.*, Avalanche characteristics of substitutionpermutation encryption networks, // IEEE Transactions on Computers, Vol. 44, No. 9, pp. 1131–1139, September 1995.

[22] *Huffman D.A.* A method for the construction of minimum redundancy codes. // In Proceedings of the Institute of Electrical and Radio Engineers 40, 9(Sept.), P. 1098 –1101. 1952.

[23] *Kam J.B. and Davida G.I.*, Structured design of substitution-permutation encryption networks, // IEEE Transactions on Computers, Vol. C-28, No. 10, pp. 747–753, October 1979.

[24] *K. Kim.* A Stady of the constraction and Analysis of substetution Boxes for Symmetric Cryptosystems: Dissert. ... Doct. Ph., Yokohama National Univeristy. Division of Electrical and Computer Engineering. 1990.

[25] *K. Kim*, Construction of DES-Like S-Boxes Based on Boolean Functions Satisfying the SAC, // Advances in Cryptology, ASIACRYPT 91 Proceedings, Springer-Verlag, 1993, P. 59-72.

[26] *Lenstra A.K., Verheul E.R.* ”Selecting Cryptographic Key Sizes”. // Journal of Cryptology. V. 14, N. 4, P. 255 –293, 2001.

[27] *Lloyd S.* Counting functions satisfying A Higher order strict avalanche criterion // EUROCRYPT 1989: P. 63 –74

[28] *Luby M. and Rackoff. C.* How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2): pp. 373–386, 1988.

[29] *Massey J.L.*, ”An Introduction to Contemporary Cryptology”, // Proc. IEEE, Vol. 76, No. 5, pp. 533-549, May 1988.

[30] *Massey J.L.* Some applications of source coding in cryptography // European Trans. On Telecomm. V. 5, P. 421-429. 1994.

[31] *Massey J.L.* On the Optimality of SAFER+ Diffusion // Proceedings of the Second AES Candidate Conference, NIST, Mar. 1999.

[32] *M. Matsui*, Linear cryptanalysis method for DES cipher, // Advances in Cryptology: Proceedings of EUROCRYPT’93, SpringerVerlag, Berlin, pp. 386–397, 1994.

[33] *Maurer U.L., Massey J.L.* Casade Ciphers: The Importance of Being First. // Jornal of Cryptology, V.6, N. 1 P. 55 –61, 1993.

[34] *Memnezes A., van Oorshot P., Vanstone S.* Handbook of applied cryptography. CRC Press, 1996

[35] *Michtchenko V.A.* Method of Encryption, Transmitting and Decrypting of Information in Public Networks // Application WO 02/091667 A1 14 November 2002.

[36] *Mischenko V.A., Zakharau U.U., Vilansky Y.V., Verzhbalovich D.I.* Method for encrypting information and device for realization of the method. // International Application Number: PCT/BY99/00005. International Publication Number: WO 00/65767. International Publication Date: 02 November 2000. Int. Filing Date: 16 Mart 1999. http://pctgazette.wipo.int/. 23 p. (7 p.)

[37] *Mischenko V.A., Zakharau U.U., Vilansky Y.V., Verzhbalovich D.I.* Method for encrypting information and device for realization of the method. International Application Number: PCT/BY99/00005. International Publication Number: WO 00/65767. International Publication Date: 02 November 2000. Int. Filing Date: 16 Mart 1999. http://pctgazette.wipo.int/

[38] *Mischenko V.A., Zakharau U.U., Vilansky Y.V.* Methods for encoding, decoding, transferring, storage and control of information, systems for carrying out the methods. // International Application Number: PCT/BY99/00008. International Publication Number: WO 01/30017. International Publication Date: 26 April 2001. Int. Filing Date: 15 October 1999. http://pctgazette.wipo.int/. 105 p. (35 p.)

[39] *Mischenko V.A.* Method for authorized displaying information distributed through public communication media. // International Application Number: PCT/BY01/00013. International ublication Number: WO 03/017566. International Publication Date: 27.02.2003. Int. Filing Date: 20.08.2001.

[40] *Mischenko V.A.* Method of encrypting, transmitting and decrypting of information in public networks. // International Application Number: PCT/BY01/00006. International Publication Number: WO 02/091667. International Publication Date: 14.11.2002. Int. Filing Date: 04.05.2001.

[41] National Institute of Standards and Technology, NIST FIPS PUB 197, Advanced Encryption Standard. U.S. Department of Commerce. 2001.

[42] New European Schemes for Signatures, Integrity, and Encryption (NESSIE). Website https://www.cryptonessie.org.

[43] *Nyberg. K.* Perfect nonlinear S-boxes. In D. Davies, editor, *Advances in Cryptology – Eurocrypt'91: Workshop on the Theory and Application of Cryptographic Techniques, Brighton, UK, April 1991. Proceedings, volume 547 of Lecture Notes in Computer Science*, pages 378–386. Springer-Verlag, 1991.

[44] *O'Connor L.*, Enumerating nondegenerate permutations, Technical Report 2527, University of Waterloo, Waterloo, Ontario, Canada, 1991.

[45] *O'Connor L.*, On the distribution of characteristics in bijective mappings. // In T. Helleseth, editor, Advances in Cryptology – Euro-crypt'93: Workshop on the Theory and Application of Cryptographic Techniques, Lofthus, Norway, May 1993. Proceedings, volume 765 of Lecture Notes in Computer Science, pages 360–370. Springer-Verlag, 1993.

[46] *O'Connor L.*, "Enumerating Nondegenerate Permutations", // Advances in Cryptology–EUROCRYPT '93 Proceedings, Springer-Verlag, 1994, pp. 368-377.

[47] *O'Connor L.*, On the distribution of characteristics in composite permutations. In D. Stinson, editor, // Advances in Cryptology – Crypto'93: 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993. Proceedings, volume 773 of Lecture Notes in Computer Science, pages 403–412. Springer- Verlag, 1994.

[48] *B. Preneel, A. Bosselaers, V. Rijmen, B. Van Rompay L. Granboulan, J. Stern, S. Murphy, M. Dichtl, P. Serf E. Bi-*

ham, O. Dunkelman, V. Furman F. Koeune, G. Piret, J-J. Quisquater, L. Knudsen, H. Raddum Comments by the NESSIE Project on the AES Finalists. [Electronic resource]. 2000. Mode of access: http:// csrc.nist.gov/ CryptoToolkit/ aes/round2/comments/20000524-bpreneel.pdf

[49] *Rivest R.L.* All-Or-Nothing Encryption and The Package Transform // Fast Software Encryption 1997, LNCS 1267,. Springer-Verlag, 1997, P. 210-218.

[50] *Rivest R.L.*, "The MD5 Message Digest Algorithm", RFC 1321, Apt 1992.

[51] *Shannon C.E.* A mathematical theory of communication. // The Bell System Technical Journal, Vol. 27, P. 379 –423, 1948.

[52] *Shannon C.E.* Communication theory of secrecy systems, // The Bell System Technical Journal, 28 (1949), P. 656 –715.

[53] *A.F. Webster and S.E. Tavares*, "On the Design of S-Boxes", // Advances in Cryptology–CRYPTO '85 Proceedings, v. 219, Lecture Notes in Computer Science, Springer-Verlag, 1986, pp. 523-534.

[54] *A.M. Youssef and S.E. Tavares.* Linear approximations of injective S-boxes. Electronic Letters, 31(25):2165–2166, 1995.

[55] *A.M. Youssef and S.E. Tavares.* Number of nonlinear regular S-boxes. Electronic Letters, 31(19):1643–1644, 1995.

[56] *A.M. Youssef and S.E. Tavares*, Resistance of balanced s-boxes to linear and differential cryptanalysis, // Information Processing Letters, Vol. 56, pp. 249–252, 1995.

[57] *Youssef A.M., Tavares S.E., and Heys H.M.*, A new class of substitution-permutation networks, // SAC '96 — Third Annual Workshop on Selected Areas in Cryptography,

Queen’s University, Kingston, Ontario, pp. 132–147, August 1996.

[58] *Youssef A.M. and Tavares S.E.*, Information Leakage of Randomly Selected Functions // Proceedings of the 4th Canadian Workshop On Information Theory. May 28, 1995, Lecture Notes in Computer Science (LNCS 1133), Springer-Verlag, 1996. P. 41 –52.

[59] *Zhang M., Tavares S.E., and Campbell L.L.*, Information Leakage of Boolean unctions and its Ralationship to Other Cryptographic Criteria // Proceedings of 2nd ACM Conference on Computer and Communications Security, Fairfax,Virginia, 1994. P. 156 – 165.

[60] *Ziv J. and Lempel A.* A universal algorithm for sequential data compression. // IEEE Trans. Inf. Theory IT-23,3, 3 (May), P. 337-343. 1977.